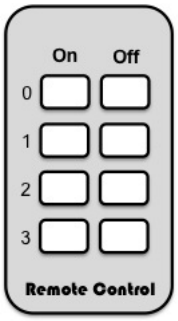
1. In der Vorlesung wurde ein dynamischer Stack in C++ implementiert. Bei einem leeren Stack liefert die Operation `pop` dort den Rückgabewert -1. Damit kann der Wert -1 nicht ordnungsgemäß auf dem Stack gespeichert werden. Ergänzen Sie die Implementierung daher um ein *Exception-Handling*. Bei einem leeren Stack soll die Operation `pop` eine selbst definierte *Exception* vom Typ `empty_stack_exception` werfen. Fügen Sie die eigene *Exception* sinnvoll in die bestehende *Exception*-Hierarchie der Standard-Library ein. Führen Sie im Hauptprogramm die Operation `pop` auf einem leeren Stack aus. Vermeiden Sie einen Programmabbruch durch eine nicht behandelte Ausnahme[1].

2. Sie implementieren Klassen für eine programmierbare Fernbedienung in der Sprache C#. Die Fernbedienung hat vier *On*-Tasten und vier *Off*-Tasten. Die Tasten sind paarweise angeordnet. Die Tasten können paarweise frei belegt (programmiert) werden.



   Schreiben Sie eine Klasse `RemoteControl` für die programmierbare Fernbedienung. Sie müssen natürlich keine GUI für die Fernbedienung programmieren. Wir steuern die Fernbedienung allein über die Methoden der Klasse `RemoteControl`. Auch die angesteuerten Geräte reagieren nur mit einfachen Textausgaben.

   Über eine Methode `void SetCommand(int i, Command on, Command off)` kann das i-te Tastenpaar mit Funktionen für ein bestimmtes Gerät belegt werden. Mit den Methoden `void PressOn(int i)` und `void PressOff(int i)` kann die i-te *On*- bzw. die i-te *Off*-Taste gedrückt werden.

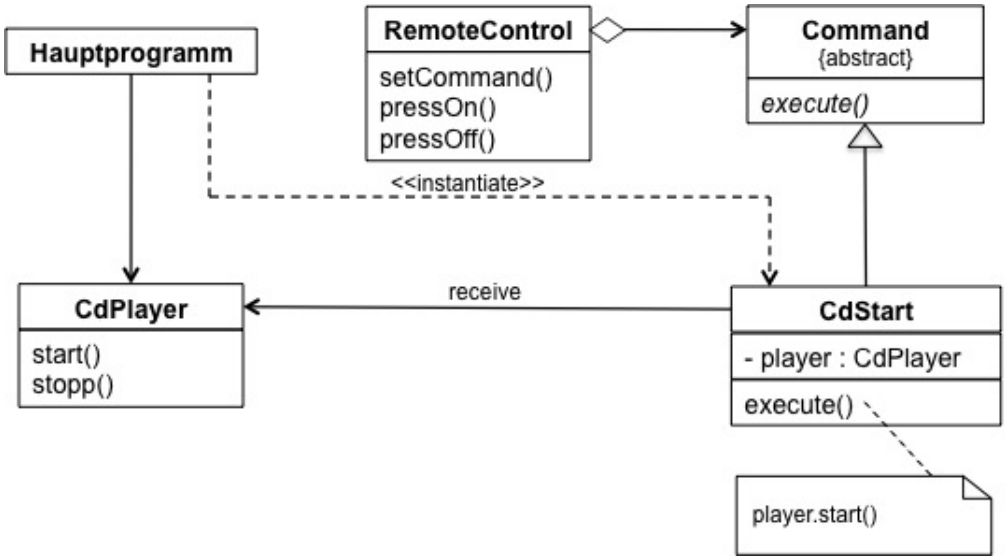
   Über die Tasten sollen unterschiedliche Geräte bedient werden können. Die Geräte werden über Klassen angesteuert. Es gibt z.B. eine Klasse `CdPlayer` mit den Methoden `void Start()` und `void Stopp()`:

```
class CdPlayer {
   public void Start(){
      Console.WriteLine("CD-Player start");
   }
```

[1] Als Zusatzaufgabe können Sie auch die Template-Version auf ein Exception-Handling umbauen.

```
   public void Stopp(){
      Console.WriteLine("CD-Player stopp");
   }
}
```

Schreiben Sie zudem eine Klasse `Garagentor` mit den Methoden `void Hoch()` und `void Runter()`. Für die Zukunft können beliebig viele weitere Geräte-Klassen auf den Markt kommen. Alle sollen dann über die Klasse `RemoteControl` angesteuert werden können, ohne dass die Klasse `RemoteControl` geändert werden muss, und ohne dass Anpassungen an den Geräteklassen erforderlich sind. Dazu entkoppeln wir die Fernbedienung über eine abstrakte `Command`-Klasse von den konkreten Geräten. Die Tasten der Fernbedienung werden dann mit konkreten Kommandos belegt. Für den Cd-Player müssen also zwei konkrete Kommandos implementiert werden. Jedes Cd-Player Kommando muss den konkreten CD-Player kennen, um dort die richtige Taste „drücken“ zu können. Diese Situation ist für die Taste *Start* des CD-Players im folgenden Klassendiagramm skizziert.



Das Klassendiagramm ist in der allgemeinen UML-Notation angegeben. Bitte verwenden Sie für die Methodennamen die empfohlene C#-Schreibweise. Die Signaturen der Methoden sind im Diagramm absichtlich unvollständig angegeben. Wählen Sie für die Implementierung geeignete Signaturen. Erstellen Sie die vier Namensräume `Fh.Pk2.Devices`, `Fh.Pk2.Commands`, `Fh.Pk2.Rc` und `Praktikum13`. Die Namensräume enthalten in der angegebenen Reihenfolge die konkreten Geräte, die Kommandos, die Fernbedienung (inkl. dem abstrakten Kommando) und das Hauptprogramm. Erstellen Sie für jeden Namensraum eine eigene Datei.

Schreiben Sie ein Hauptprogramm. Dort belegen Sie die Tasten der Fernbedienung,

so dass Sie einen Cd-Player und ein Garagentor über die Fernbedienung steuern können. Erzeugen Sie über die Fernbedienung die folgende Ausgabe:

```
CD-Player start
Garagentor hoch
CD-Player stopp
Garagentor runter
```

Zur weiteren Übung können Sie `Command` als Interface implementieren. Zusätzlich können Sie die Aufgabe auch mit C++ implementieren.