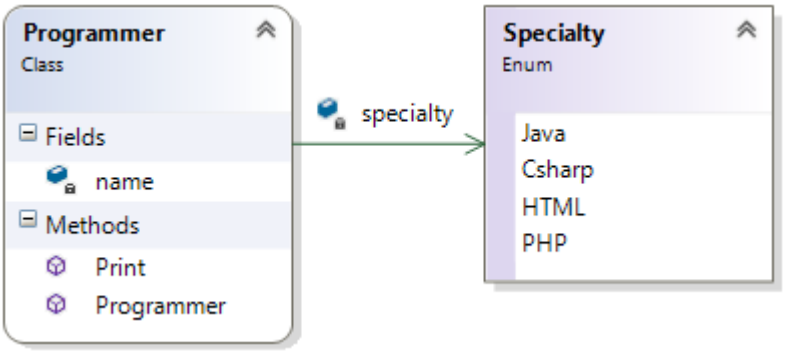
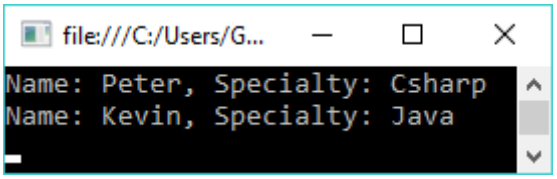
Opdracht 1 – Softwarebedrijf

Het softwarebedrijf Blue Pigeon heeft een deel van het personeel ingedeeld in teams. Elk team bestaat uit vier programmeurs. Elke programmeur heeft een specialiteit, bijvoorbeeld Java, C#, HTML of PHP.

a) Maak een enumeratie *Specialty* die deze specialiteiten bevat, en sla deze enumeratie op in een apart bestand.

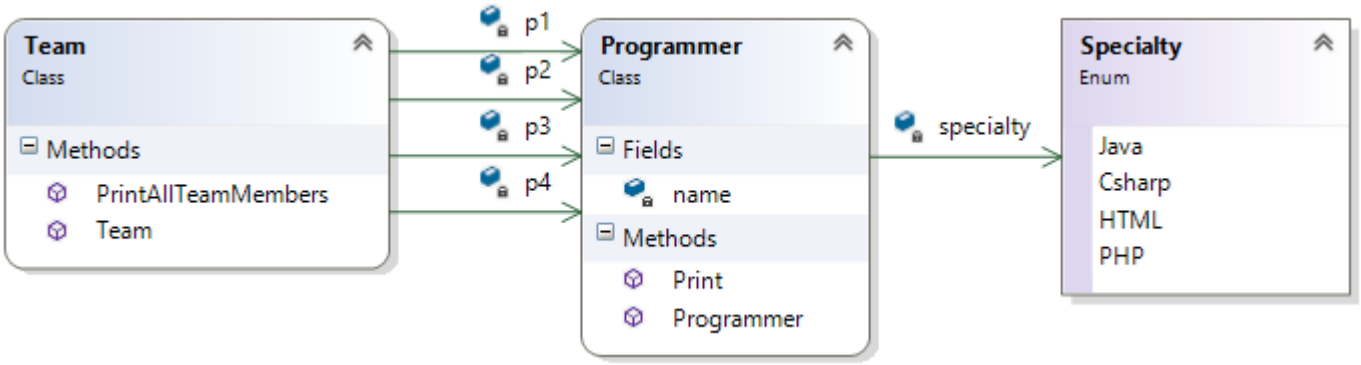
b) Maak een class *Programmer* (in een apart bestand) die een naam bevat en een member van het type *Specialty*. Geef deze class een constructor waarin je de naam van de programmeur en zijn/haar specialiteit kunt meegeven. Geef de class ook een methode Print() die de velden van het betreffende object op het scherm zet. Test deze Print-methode door een aantal programmeurs aan te maken en te printen.

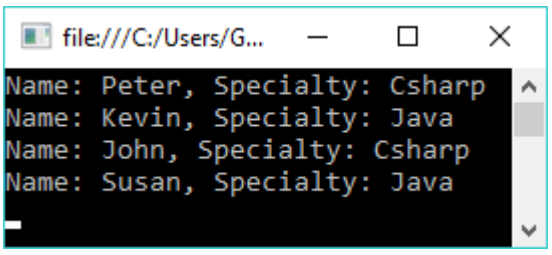




Voorbeeld van Print (class Programmer)

c) Maak een class *Team* (in een apart bestand) die in de constructor vier nieuwe instanties van *Programmer* maakt met zelfverzonnen namen en specialiteiten. Geef de class een methode *PrintAllTeamMembers()* die de namen van de programmeurs en hun specialiteit op het scherm zet. Roep deze methode vanuit de Start-methode aan om het resultaat te laten zien (maak hiervoor eerst een Team object aan).





Voorbeeld van PrintAllTeamMembers (class Team)

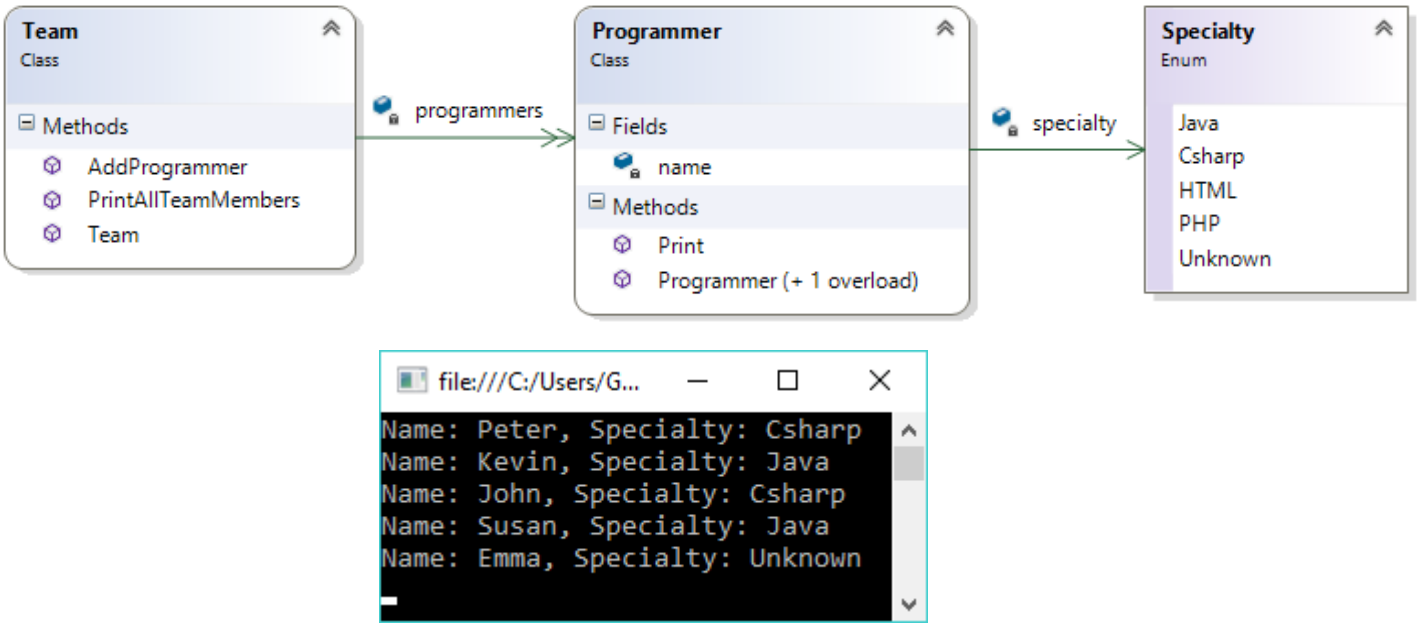
Denk nu eens na over dit ontwerp; is het praktisch/bruikbaar?

d) Wijzig class Team: voeg een lijst van programmeurs toe, wijzig de constructor (maak hier de lijst van programmeurs aan), en voeg een methode AddProgrammer toe die er als volgt uit ziet:

```
public void AddProgrammer(Programmer p)
{
    // add 'programmer' to the list
    // ...
}
```

Wijzig de Start-methode: het aanmaken van de Programmeur-objecten (instanties) zal nu niet meer gebeuren in class Team (zoals gevraagd bij c), maar kan nu in de Start-methode plaatsvinden. Dit zorgt er voor dat class Team niet altijd dezelfde (hardcoded) programmeurs bevat, maar dat het team samengesteld kan worden door de 'gebruiker' van deze class.

e) Voeg aan class Programmer een 2e constructor toe die alleen de naam van de programmeur ontvangt, en de specialiteit op 'Unknown' instelt. Je kunt hierbij de andere constructor (met 2 parameters) aanroepen (zie §4.7.4 v/h Yellow-book). Test deze 2e constructor door een programmeur zonder specialiteit toe te voegen aan het team, en vervolgens het team weer te printen.



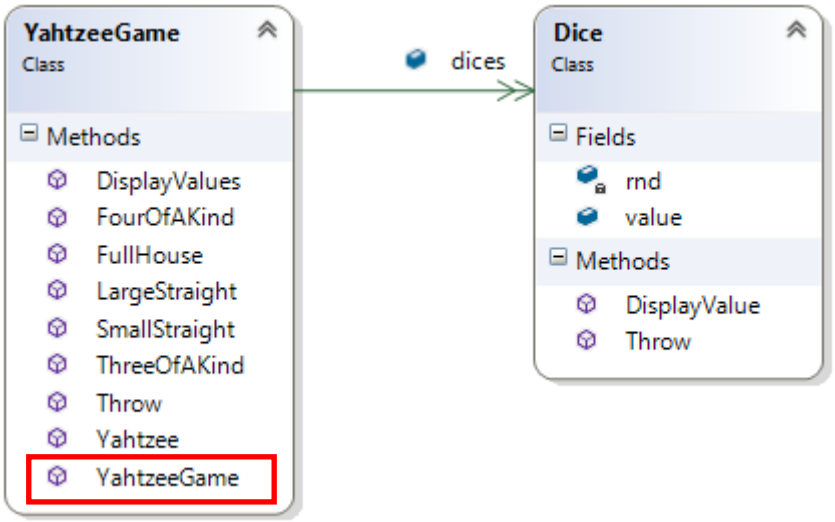
Voorbeeld van PrintAllTeamMembers (class Team), waarbij een programmeur zonder specialiteit is aangemaakt (en toegevoegd is aan het team)

f) Controleer of de classdiagram van jouw eigen VisualStudio-project hetzelfde is als hierboven. Gebruik hiervoor "View Class Diagram" in Visual Studio. Om de pijlen te krijgen, klik op een veldnaam en selecteer popup menu-item "Show as Association" (voor gewone velden zoals *specialty*) of "Show as Collection Association" (voor lijsten zoals *programmers*).

Opdracht 2 – Yahtzee game

Nu we weten dat er een speciale methode is om een object te initialiseren (genaamd 'constructor'), kunnen we class YahtzeeGame uit week 1 van Programmeren 2 aanpassen.

a) Verwijder de Init-methode en voeg een constructor toe waarin de 5 dobbelstenen worden aangemaakt. Verwijder natuurlijk ook de aanroep van de Init-methode (gebeurt vanuit de Start-methode).
→ Controleer of de Yahtzee-applicatie nog werkt (bv met de loop die wacht totdat er Yahtzee is gegooid, 5 gelijke dobbelstenen).



Zoals je kunt zien, heeft class YahtzeeGame geen Init-methode meer (maar een constructor)

b) Class 'Dobbelsteen' (Dice) heeft een static random generator, hier willen we van af. Wat gebeurt er als je 'static' verwijderd en het programma nogmaals runt? Krijg je nu ook steeds in 1x Yahtzee?

Dit komt omdat zonder 'static' elke Dobbelsteen een eigen Random-object aanmaakt, en omdat de (5) Dobbelsteen-objecten vlak na elkaar worden aangemaakt, worden de Random-objecten ook vlak na elkaar aangemaakt. Een Random-object gebruikt de huidige tijd om 'pseudo-random' getallen te genereren, en omdat de huidige tijd voor alle (5) objecten gelijk is, zijn ook de gegenereerde getallen steeds gelijk.

c) Geef class Dobbelsteen een constructor met als parameter een Random-object, en zorg er voor dat deze 'intern' (in een member) wordt opgeslagen, zodat de methode 'Gooi' er gebruik van kan maken. Pas de constructor van class YahtzeeGame aan zodat daar één Random-object wordt aangemaakt, en deze wordt doorgegeven aan alle Dobbelsteen-objecten (tijdens het aanmaken).
→ Controleer of de applicatie weer normaal werkt.