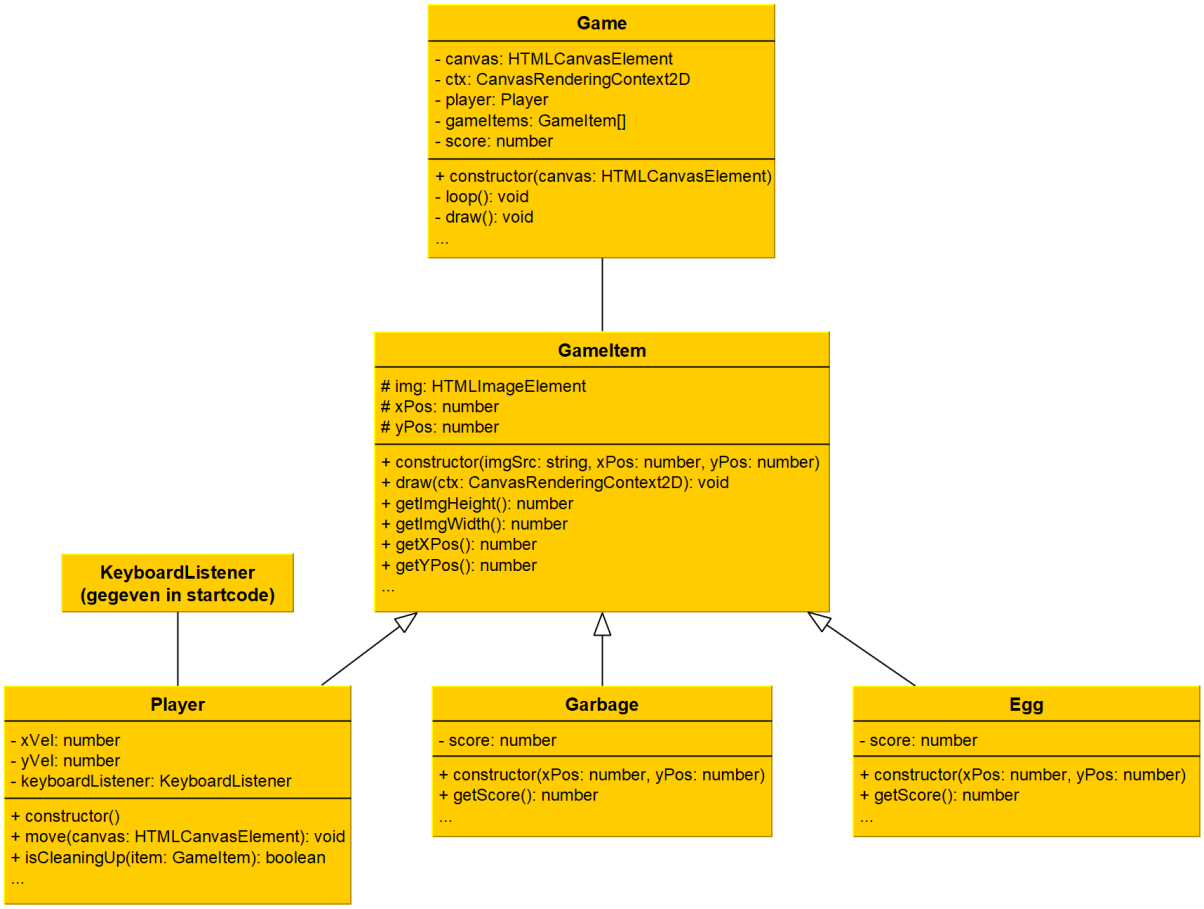
Een vriend van je is pas aangenomen bij een software-ontwikkelbedrijf. Dit bedrijf wil zich beter in de markt zetten door het maken van serious games voor scholieren. Echter, niemand in het bedrijf heeft ervaring met het maken van games. Daarom heeft jouw vriend de opdracht gegeven uit te zoeken hoe game development werkt, een eerste game te bedenken en uit te werken. Dat is gelukt, en je vriend lijkt een game gemaakt te hebben die het bedrijf graag wil verkopen. Er moet alleen nog een scoresysteem worden toegevoegd. Maar daar had je vriend geen rekening mee gehouden. Het toevoegen van de tekst om de score weer te geven is gelukt, maar elke poging om ook maar iets van scorefunctionaliteit aan de game toe te voegen mislukt hopeloos.

Gelukkig heb jij de cursus Object Oriented Programming gevolgd en kan jij wel bedenken wat hier aan de hand is. Je opent de code en inderdaad: vrijwel alles staat in één klasse waarin alles dus afhankelijk is van alles. Na een half uurtje tekenen en nadenken, ben je tot het volgende ontwerp gekomen.



Nu belt je vriend in paniek op: de game moet morgen af zijn. Hij belooft je de helft van zijn salaris als je kan helpen. Kun je het herontwerp implementeren en het scoresysteem gelijk toevoegen?

De game

"Super cleaner" is een game waarin de speler rondloopt in een weiland, met daarin de taak om zoveel mogelijk zwerfafval op te ruimen. Maar let daarbij wel op dat je geen vogeleieren weghaalt, die moet je natuurlijk laten liggen! Het spel in in principe oneindig: op willekeurige tijdstippen verschijnen er nieuw afval en vogeleieren.

Je kan de speler besturen met de pijltjestoetsen. Als je karakter op een stuk zwerfafval staat, kan je dat opruimen door op spatiebalk te drukken. Zwerfafval wordt in "Super cleaner" weergegeven door een smeltend ijsje.

Toetsopdracht

Schrijf de gegeven code om volgens het gegeven klassediagram. Voeg daarbij scorefunctionaliteit toe zoals beschreven in het klassediagram. De score in de game wordt berekend aan de hand van de volgende regels:

- Het opruimen van afval levert 1 punt op
- Het opruimen van een vogeleieren zorgt voor 5 punten aftrek
- De score kan negatief worden

LET OP: het kan gebeuren dat je tijdens je implementatie andere keuzes maakt dan het klassediagram voorschrijft. Dit wordt aangegeven met de drie puntjes in het klassediagram. Als het gebeurt dat je implementatie afwijkt van het klassediagram, leidt dat – mits je uitwerking voldoet aan de beoordelingscriteria – niet tot een slechtere beoordeling.

Zie ommezijde voor handige tips!

Tips:

- Het “npm run watch” commando gaat niet altijd goed om met nieuw toegevoegde klassen. Als je een nieuwe klasse toevoegt, herstart dan de compiler (in de terminal: “ctrl + c” en daarna weer “npm run watch”).
- Refactor niet te veel code tegelijk. Controleer na elke refactoring of je game nog doet wat die moet doen.
- Kijk goed naar de verschillen in constructor van de klasse GameItem en zijn subklassen in het klassediagram.
- Tijdens het implementeren van het opruimen van GameItems, loop je waarschijnlijk tegen een onhandige ontwerpkeuze aan. Hoe weet de compiler het verschil tussen een Player, Egg en Garbage? Je mag daarvoor het instanceof commando gebruiken en een check op Egg of Garbage zetten zodat je de Player omzeilt:

  ```
  if (element instanceof Egg || element instanceof Garbage) {
      // Clean up element
  }
  ```

  Je mag het probleem uiteraard ook op een betere manier oplossen, maar dan zal je moeten afwijken van het klassediagram.

Inleveren

Je levert een .zip bestand in (geen .rar, .7z, of ander formaat) van maximaal 5mb. Dat .zip bestand bevat uitsluitend de “src/” map met alle code. Alle andere mappen of bestanden lever je niet in.

Beoordelingscriteria

Code die fundamenteel niet compileert, levert geen punten op. Dat wil zeggen, een enkele spelfout of verkeerd haakje wordt door de beoordelaar opgelost, waarna je uitwerking wordt nagekeken. Code waarvoor langer dan een halve minuut moet worden opgelost door de beoordelaar wordt beschouwd als “compileert fundamenteel niet”.

Nummer	Leeruitkomst	Aantal punten
1	Je pas op consistente manier indentatie in de code toe	0,2
2	Je voorziet de code volgens een standaardafspraak – AirBnB JavaScript Style Guide – van commentaar	0,4
3	Je geeft consistent betekenisvolle namen aan variabelen, methodes en klassen volgens AirBnB JavaScript Style Guide	0,4
4	Je structureert je code in klassen met attributen en methodes	2
5	Je pas overerving toe om codeduplicatie te voorkomen	2
6	Je geeft datatypes van attributen en parameters, en returnwaarden van methodes expliciet aan. Datatype “any” is niet toegestaan.	2
7	Je pas encapsulation toe in een klasse zodat de data van die klasse op een goede manier wordt weergegeven voor zijn omgeving	1
8	Je past polymorphisme toe opdat code minimaal wordt herhaald	1
9	Je past DRY-principes toe zodat de complexiteit van de code wordt verdeeld in onderhoudbare onderdelen	1