Oefeningen – Collections

Oefening 1

Maak een klasse ***Card,*** die een speelkaart zal voorstellen. De klasse heeft member variabelen *suit* en *value*. Beide variabelen zijn van een *enum* -type:

- enum Suit: CLUBS, DIAMONDS, HEARTS, SPADES
- enum Value: TWO, THREE, FOUR, FIVE, …, KING, ACE

Deze enums krijg je al cadeau.

Gebruik zo veel mogelijk functionaliteit uit deze enums.

Je kan ook het UML diagram op de laatste pagina gebruiken.

Maak een klasse ***Deck*** die in de constructor een volledig kaartspel aanmaakt en in een Queue steekt. Zorg er voor dat de kaarten onmiddellijk daarna geschud worden. (Tip: Bekijk de klasse Collections, misschien vind je wel een manier om de kaarten gemakkelijk te schudden). Deze klasse stelt dus een willekeurig geschud stapeltje kaarten voor, waarvan we telkens enkel de bovenste kunnen opvragen door een Queue te gebruiken.

Maak daarna een methode ***dealCard()*** die de bovenste kaart van de stapel zal verwijderen en teruggeven.

De *Deck* klasse heeft ook een methode ***getSize()***, die op elk moment het resterende aantal kaarten in het deck terug geeft.

We willen nu een klasse ***Player*** waaraan we kaarten kunnen delen. Voorzie een member variabele *hand* waarin deze kaarten bijgehouden kunnen worden. De kaarten in de hand moeten steeds gesorteerd zijn. Zoek zelf uit welke Collection je hiervoor best kan gebruiken, als type voor *hand*.

Het sorteeralgoritme moet als volgt werken: eerst moeten alle harten (hearts) zitten, daarna ruiten (diamonds) , klaveren (clubs) en tenslotte schoppen (spades). De kaarten van dezelfde 'kleur' (= harten, klaveren, ruiten of schoppen) zijn gesorteerd volgens oplopende waarde: 2, 3, …, J, Q, K, A.

Maak een methode ***addCard()*** die een kaart als parameter krijgt en deze in de gekozen Collection toevoegt. Denk er aan dat de kaarten in je hand (en dus in de Collection) steeds gesorteerd moeten blijven. Als je de juiste Collection hebt gekozen, gebeurt dit automatisch wanneer je een Compare methode hebt gedefinieerd voor de elementen in de Collection.

Maak een methode ***hasSuit()*** die een *Suit* als parameter mee krijgt. Deze functie geeft een boolean waarde terug; *true* als er ten minste 1 kaart met de meegegeven kleur in de hand zit, anders *false*.

Schrijf de ***toString()*** methode die de kaarten in de hand – in gesorteerde volgorde – als een String terug geeft. Een kaart print je als volgt: Klaveren 6 wordt "6 of CLUBS", Ruiten boer (jack) wordt "JACK of DIAMONDS", Schoppen koningin (queen) wordt "QUEEN of SPADES", Harten koning wordt "KING of HEARTS", etc.
De kaarten worden in deze functie gewoon achter elkaar geplakt in een String. Voorgaande vier kaarten zouden dus getoond moeten worden als "KING of HEARTS JACK of DIAMONDS 6 of CLUBS QUEEN of SPADES" . Maak gebruik van streams waar dat handig kan zijn.

Voorzie tenslotte ook in de Deck klasse een ***toString()*** methode, die de hele inhoud van het deck toont, op dezelfde manier als een Hand geprint kan worden. (maar dan niet gesorteerd, want de kaarten in de *deck* zijn random geschud).

Test je resultaat met de CardTest.

Oefening 2

Maak een enum *Coin* met alle mogelijke euromunten (1 cent, 2 cent, etc.).

Maak een klasse Portemonnee met daarin een ***Map*** waarin we voor elke munt het aantal bewaren. Voeg methodes toe om munten toe te voegen en te verwijderen.

Maak ook een methode om het totaalbedrag van de portemonnee op te vragen en een toString methode die de inhoud van de portemonnee afdrukt. Test de werking in je hoofdprogramma.

UML oefening Cards