Uppgift 2 - Spelet Nm

Ville Wassberg
Ville.wassberg@gmail.com

February 2023

Köra programmet

Den som ska köra programmet kan starta det genom att stå i mappen Inlämningsuppgift2VilleWassberg/NmSpel
där man nu väljer att spara den, och skriva:

*javac *.java*

sedan

java Nm

.
Tanken är då att användaren får välja namn på spelare ett och spelare två, samt svara på hur många stickor man vill att högen ska bestå av. Sedan dyker en dialogruta upp där man får klicka i vilken sorts spelartyp man vill att spelarna ska vara. Nu startar spelet!

Rapport

Programmet för spelet Nm är indelat i 5 olika klasser. En huvudklass som heter som spelet; Nm, där main-metoden körs. Förutom main så har Nm flera viktiga metoder som bland annat definierar spelets semantik, samt en metod som är till för att skapa de spelartyper som användaren vill att spelarna ska var (Human eller Computer). En klass, Pile, som representerar högen med stickor, finns för att konstruera högen, samt hålla koll på antalet stickor i den och för att kunna göra förändringar i högen när någon spelare vill plocka bort stickor från den. Jag har även använt en abstrakt klass (som jag i ärlighetens namn inte helt har landat i varför den bör vara abstrakt) Player, som är en superklass till två klasser som ska representera spelarna som väljs. Player instansierar de objekt och metoder samt konstruktorer som jag tyckte bör vara gemensamma för alla spelare som kan komma att spela Nm. Exempelvis valde jag att ge spelarna ett namn, vilket alla spelare fick, även om datorn kanske kunde ha ett cpu-namn hela tiden, men jag tyckte det var roligare att användaren får välja. Computer är en klass som representerar en spelare som varken är särskilt bra eller dålig då den väljer ett tal i det givna intervallet helt slumpartat. Human är en klass

som har en speciellt viktig metod (likt Computer), vilket är den som ska ta reda på och returnera hur många stickor spelaren vill plocka bort från högen.

Jag kom fram till min lösning genom att dela upp programmet just i dessa klasser. Jag började med att göra en lista om objekt som jag tyckte borde finnas. Jag delade upp vad varje objekts uppgift i spelet borde vara, vilket jag tänkte skulle få bli deras metoder. Därefter kändes det relativt klart vad som skulle vara var i programmet.

Jag började med att tänka på stickorna som enskilda objekt som kanske även dem skulle få sin egna klass, men jag övertygade mej själv till att låta bli, därför att det kändes mer naturligt att endast låta det vara ett av heltalen i högen.

I efterhand, är jag ändå relativt nöjd med de designbeslut jag tog. Dock tycker jag ändå att det ser lite stökigt ut. Jag kanske kunde ha skapat en metod som enbart har i syfte att läsa in heltal och strängar ifån användaren. Möjligen till och med uppdelade i egna klasser, då det blev lite klumpigt med alla try-catch och scanner-metoder. Möjligen borde jag kanske ha följt exemplet ReadInteger, som vi hade till hands. Jag började med att motivera mitt användande av JoptionPane med att jag inte skulle behöva try-catch, men ändå slutade jag upp med att läsa in namn för att fånga upp via try-catch, vilket såhär i efterhand känns lite onödigt; däremot var det lärorikt att testa mej fram såpass mycket som jag gjorde. Jag kunde även ha gjort något åt att när man splar spelet som bara datorer så printas alla drag i stort sett bara ut på skärmen. Det hade nog varit trevligare för användaren att kanske kunna klicka sig igenom dragen, exempelvis klicka enter när den vill att nästa spelare ska utföra sitt drag.

Komplettering

Nu när jag körde programmet själv, igen, efter att ha blivit ombedd att komplettera tedde det sig inte alls så som jag vill minnas att det gjorde före inlämningen, möjligen missade jag att göra något sparande under stressen när jag insåg att inlämningen var försenad. Jag började med att titta på while-loopen i Human-klassen, där jag insåg att jag hade loopen runtom mina try-catch-satser, när den borde ha varit under try-satsen; så den förändringen gjorde jag. Vid en testkörning så fungerar interagerandet med programmet som tänkt och instruerat, både med human-human, human-computer, samt computer-computer.

Jag ändrade också utskriften av vem som vinner till det korrekta, så att den som har en sticka kvar vid sin tur förlorar, alltså att spelaren innan vinner, vet inte riktigt hur jag misstolkade det från början. Dock vet jag inte varför det bara gick att mata in 1 sticka för testaren, för mej har det fungerat att göra olika input även innan dessa ändringar, men förhoppningsvis fungerar det bättre nu. Jag har även tagit bort alla static metoder och variabler, som jag tror att jag satte som static därför att jag fick fullt av error-meddelanden där det stod ”

non-static method/variable cannot be referenced from a static context", när jag försökte köra programmet. Jag skapade istället en konstruktor till Nm (som jag visst hade missat) så jag plockade helt enkelt ut innehållet i main-metoden och stoppade in det i den, tog bort all användning av static och anropar konstruktorn i main istället. Det kändes som den enklaste lösningen.

Jag gjorde även ändringar i try-catch i human-klassen, därför att jag inte dirket fångade några miss inputs på något vettigt sätt; nu använde jag mej av rekursion och skickade om human om det blev ett input exception som fel typ på nummer eller om man skickar in en sträng tillexempel. Jag gjorde även texten mer lättläslig i terminalen genom att låta varja spelartur bli ett stycke istället för att allt var sammanhängande. Jag följde även ett råd från min peer review så att användaren kan ha namn med mellanslag. Hoppas det funkar fint!