Leertaak 2: Project Vossen & Konijnen

Tijdens deze leertaak ga je in een projectgroep van minimaal drie, maximaal vier personen werken aan het ontwikkelen van een wat grotere applicatie. De basis is het project Vossen en Konijnen dat je al hebt leren kennen bij het practicum (hfst 10). Tijdens dit project komen alle onderwerpen van dit thema bij elkaar (Java programmeren, ontwerpen met UML, versiebeheer en tooling, het maken van een GUI, etc).

N.B. Voor alle opdrachten geldt dat je individueel moet kunnen uitleggen hoe iets werkt. De verschillende tools (Eclipse, SVN, JUnit, debugger) moet je beheersen. Het kunnen werken met deze tools kan deel uitmaken van een individuele beoordeling aan het eind van het project.

(zie volgende bladzijde)

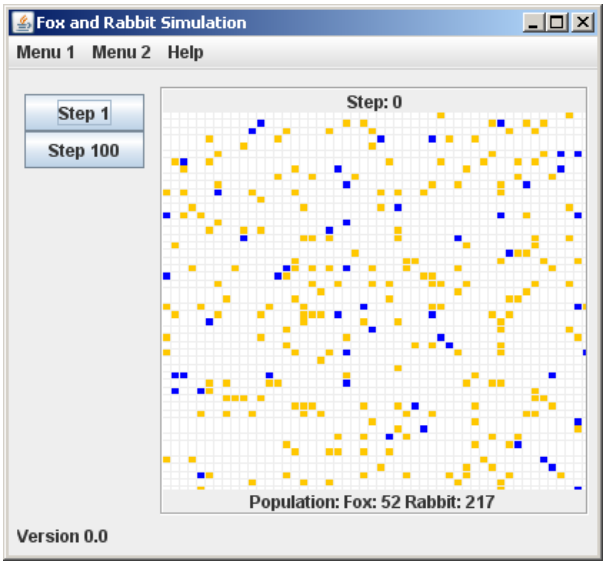
Periodeweek 7

Eclipse
Als je nog niet gewerkt hebt met Eclipse, download en installeer het dan. Op Blackboard staat informatie over de overgang van BlueJ naar Eclipse (BB->Ondersteunende informatie->Tutoraat).
Oefen wat met Eclipse en importeer het project foxes-and-rabbits-v2.

SVN repository
Als je nog niet gewerkt hebt met SVN, maak dan een repository aan en installeer TortoiseSVN en de plugin Subclipse. Zie de aanwijzingen aan het begin van leertaak 1 en de aanwijzingen op Blackboard->Ondersteunende informatie->Tutoraat.
Zet de begincode van jullie project in jullie gezamenlijke repository. Zet vanaf nu alle wijzigingen aan de code in SVN.

Main methode
Voor het toepassen van een programma buiten BlueJ wordt gebruikt gemaakt van de methode "main". Bestudeer bijlage E "Java uitvoeren zonder BlueJ" die beschrijft hoe je een main-methode maakt en hoe je een *.jar file maakt. Implementeer die main-methode in je projectcode.

Eenvoudige GUI
Ontwikkel een eenvoudige GUI (zie voorbeeld) met knoppen voor 1 stap en 100 stappen, zodat je de vossen en konijnen simulatie kunt uitproberen. Steek hier niet teveel tijd in, de volgende weken ga je de GUI aanpassen en uitbreiden. Kijk ook eens naar de vijf Eclipse projecten die in mvcextended.zip zitten, te vinden op Blackboard. Je ziet daar hoe je de applicatielogica en de GUI netjes uit elkaar wordt gehouden. Volgende week ga je echt naar de MVC structuur kijken die in de vijf projecten wordt gebruikt



Figuur 1: Eenvoudige GUI

Junit testen
Maak in Eclipse een aantal unittests aan. Maak in je test bijvoorbeeld een (klein) veld aan met een (beperkt) aantal konijnen en test dat die zich op de juiste manier voortplanten. Of maak een veld aan met een aantal vossen en konijnen en test dat de konijnen op de juiste manier worden opgegeten. Zie het BlueJ boek voor uitleg over JUnit en http://www.vogella.de/articles/JUnit/article.html voor meer details. (EasyMock hoef je niet te bekijken.)

Debuggen
Gebruik de debugger in Eclipse om een paar keer door een run van vossen en konijnen heen te lopen. Experimenteer met breakpoints, step into, step over en het opvragen van variabelen.

Probleemanalyse beschrijven in rapport
Maak een opzet van je eindrapport en beschrijf daarin de volgende hoofdstukken.

1 – Probleemstelling.
Hierin dienen jullie het probleem te beschrijven dat jullie met het maken van dit programma proberen op te lossen. Probeer de probleemstelling generiek (algemeen) en realistisch te formuleren. Dus niet alleen voor vossen en konijnen, maar bijvoorbeeld ook de visstand op de Noordzee of de dierenpopulatie van een eiland in de stille Zuidzee.

2 – Analyse van de huidige situatie.
Maak een analyse van de basisversie van jullie programma (foxes-and-rabbits-v2). Wat zijn de problemen, wat zijn de verbetermogelijkheden en aan welke functionaliteit heeft de opdrachtgever behoefte?
Vragen die je jezelf kunt stellen zijn bijvoorbeeld:
- In hoeverre geven de verschillende runs van het programma verschillende dan wel vergelijkbare uitkomsten?
- Zijn er trends waarneembaar?
- Zijn er overeenkomsten met het van de economie afkomstige principe van de varkenscyclus (googelen!).
- Hoe onderhoudbaar/uitbreidbaar is de simulatie? Wat zijn de beperkingen?

Werk dit eerste hoofdstuk van jullie verslag netjes uit en stuur dit 24 uur voor de volgende afspraak met je tutor naar hem toe.

Periodeweek 8

Ontwerp
Maak een UML klassediagram van je programma. Teken een UML sequencediagram van de werking van het programma wanneer de simulatie 1 stap draait. Neem deze figuren in je rapport op in "Hoofdstuk 3 – Uitbreiding 1".

Verbeteringen aan het programma
Voeg de volgende verbeteringen en uitbreidingen toe aan je programma (opdrachten 10.38-10.51 uit BlueJ):

- Implementeer de abstracte superklasse Animal. Onderzoek welke methodes abstract kunnen worden en maak die abstract.
- Implementeer de interface Actor.
- Voeg nog een diersoort toe (wolven, beren, etc.). Bepaal van die nieuwe diersoorten of ze prooi- of jachtdieren zijn. Als deze diersoort zowel vossen als konijnen opeet dan is de vos dus een prooi én een roofdier geworden. Hoe voorkom je het "uitsterven" van diersoorten?
- Voeg een jager toe. Hoe voorkom je dat de jagers elkaar afschieten?
- Maak gebruik van testklassen of nog beter J-Unit testen om de zelfgemaakte code te testen.

Maak ook een UML klassediagram van je programma ná deze aanpassingen.

MVC, refactoring en packagestructuur
Bestudeer de structuur van de volgende drie Eclipse projecten die je in mvcextended.zip kunt vinden:

- MVCDynamicModelThread
- MVCDynamicModelThreadGeneralized: hetzelfde voorbeeld, alleen waarbij MVC achtige aspecten zoveel mogelijk in abstracte superklassen zitten
- Life: Een simulatieprogramma dat in de verte lijkt op Vossen en Konijnen, maar waar ook gebruik is gemaakt van bijna dezelfde MVC achtige abstracte superklassen. Overigens bevat dit project twee controllers (met opzet zo gekozen) om ook de flexibiliteit van MVC aan te tonen.

In alledrie projecten wordt gewerkt met Threading om er voor te zorgen dat je applicatie nog reageert op knoppen tijdens een simulatie. Threading is tweedejaarsstof en hoef je nog niet te helemaal te snappen. Je kunt er wel je voordeel mee doen in je V&K uitwerking.

Vergelijk de uitwerking van MVCDynamicModelThread met MVCDynamicModelThreadGeneralized en vergelijk de uitwerking van MVCDynamicModelThreadGeneralized met Life. Wat zijn de overeenkomsten en verschillen?

Vermeld dit in je rapport in "Hoofdstuk 4 – Het gebruik van MVC in V&K." Beschrijf nu hoe V&K de MVC structuur van Life kan krijgen in datzelfde hoofdstuk. Kies een packagestructuur zoals die wordt gebruikt in Life. Maak een ontwerp (klassediagram + sequencediagram MVC) hoe je programma er dan uit komt te zien. Laat dit ontwerp goedkeuren door je tutor en voer dan de wijzigingen door.

Maak opgaven 10.55-10.57, maar in plaats van dat je SimulatorView als interface implementeert ga je AbstractView zoals gebruikt in Life extenden met een concrete “View” subklasse.

N.B. Het belangrijkste doel van MVC is om de verschillende onderdelen van de applicatie los te koppelen van elkaar. Een goede check om te kijken of je MVC implementatie correct is, is om te kijken of je een nieuwe view of controller kunt toevoegen zónder dat je het model moet aanpassen.

Extra views
Breid het project met drie extra views, die in grafische vorm informatie over de simulatie weergeven. Bijvoorbeeld: een histogram, een cirkeldiagram en een historische weergave van de populatie (laatste 100 stappen). Geef elk dier zijn eigen kleur.

Verslaglegging
De verbeteringen en uitbreidingen die jullie deze week gemaakt hebben dienen uitgewerkt te worden in het verslag. Jullie dienen je verbeteringen en uitbreidingen toe te lichten aan de hand van het presenteren van (gedeeltes van) klassendiagrammen en stukken programmacode. Druk de (relevante) stukken programmacode verkleind af (bijvoorbeeld lettertype 8).
Onderzoek in hoeverre de ontwerpdocumentatie nog overeenstemt met de uiteindelijke implementatie. Pas eventueel de ontwerpdocumentatie aan en beargumenteer waarom je van het ontwerp bent afgeweken.

```
Programmacode illustreer je bijvoorbeeld zo…
```

Stuur je bijgewerkte rapport 24 uur voor de volgende afspraak met je mentor naar hem toe.

Periodeweek 9

Verplichte uitbreidingen
- Breid de GUI uit. Maak het mogelijk om de parameters van de diverse diersoorten in te stellen (levensduur, aantal nakomelingen, voortplantingsleeftijd, etc). Zorg ervoor dat deze parameters een default (standaard) startwaarde hebben. Overleg welke grafische componenten je gaat gebruiken. (tekstvelden, knoppen, schuifregelaars, pulldown menu’s, etc.)
- Betrek de voedselvoorraad van de prooidieren in de simulatie. Bij veel konijnen is er weinig gras en dus minder nakomelingen. Bij weinig konijnen is er veel gras en dus meer nakomelingen.
- Probeer op een zinvolle manier plaatjes en/of geluiden toe te voegen.
- Bedenk zelf (minimaal 1) zinvolle en originele aanvullingen.

Bonus uitbreidingen
- Voeg een besmettelijke ziekte toe. Konijnen hebben een (boolean) ziekteGen. Als het gen “aan” is, dan is het konijn vatbaar voor de konijnenziekte. Gemiddeld 90% (instelbaar) van de konijnen is vatbaar voor die ziekte. Als een konijn ziek is, dan is hij gedurende 5 (instelbaar) iteraties besmettelijk (voor zijn buren) daarna sterft het konijn.
  Hoe breng je de infectie op gang? Wat doet de ziekte met de andere dieren in het ecosysteem?
- Demonstreer compilatie en opstarten van jullie programma via de commandline (dus zonder een IDE).

- Indien jullie als groep nog leuke suggesties hebben om het programma aan te vullen bespreek die dan met je mentor, de implementatie van die suggesties kan ook bonus opleveren.

Beschrijf je aanpassingen in het rapport. Schrijf ook een hoofdstuk met conclusies en aanbevelingen . Je hebt nu alle 'zakelijke' inhoud voor je rapport compleet. Schrijf verder nog een stuk 'reflectie' en neem dat op in een aparte bijlage van het rapport.

Inleveren en beoordeling

Verslag

Inhoud Het verslag dient qua inhoud minimaal het onderstaande te bevatten.

- Naam project, naam van de groepsleden, inhoudsopgave en bladzijde nummering.
- Een beschrijving van de fouten en problemen die nog in je applicatie aanwezig zijn.
- Een beschrijving van je testproces (black-box, white-box en/of regressietesten)
- UML: Uitgebreid klassendiagram (incl. methoden, instantievariabelen, scope, stereotypes en static) en de belangrijkste sequencediagrammen.
- Javadoc toevoegen aan je code en de HTML pagina's via javadoc tool gegenereren.
- Wat zijn de belangrijkste (zelfbedachte) toegevoegde features.
- Geef een beschrijving op implementatieniveau (wat zijn de belangrijkste kenmerken van je implementatie, beschrijf dat bv aan de hand van begrippen als afhankelijkheid, cohesie en inkapseling en ontwerpen op basis van verantwoordelijkheden)
- Jullie zullen vast wel een keer "gerefactord" hebben. Geef daar een toelichting op.
- Een opsomming van de individuele bijdragen van elk teamlid en reflectie op het project.

Vorm Het verslag bevat alle hierboven omschreven inhoud en dient qua vorm te voldoen aan de verslagtechnische eisen zoals die in de colleges bedrijfscommunicatie aan de orde zijn gekomen. Zorg je verslag zodanig te (her)schrijven (denk aan structuur, taalgebruik en lay-out) dat het als eindrapport naar de zogenaamde opdrachtgever opgestuurd zou kunnen worden. Het makkelijkst is het als je je daartoe in de volgende situatie inleeft: Samen met de leden van je projectgroep vorm jij een ICT-bureautje NAAM dat van de (denkbeeldige) opdrachtgever de opdracht heeft gekregen de applicatie Vossen en konijnen te verbeteren.
De stukken over individuele bijdragen en reflectie horen dan dus thuis in een aparte bijlage, want in werkelijkheid zouden die stukken nooit naar een opdrachtgever gaan.

Het eindrapport is uiterlijk de dag voor de eindpresentatie om 15:00 uur in het bezit van de mentor en van de docent BBC. Je sluit het project af met een presentatie en een korte demonstratie.

Code

Vraag aan je docent hoe hij de code en javadoc wil ontvangen (mail/CD/papier). De code zal worden beoordeeld op

- correctheid
- geschikt gebruik van taalconstructies
- stijl (commentaar, inspringen, betekenisvolle naamgeving van variabelen)
- moeilijkheidsgraad en hoeveelheid functionaliteit
- originaliteit en creativiteit
- correct gebruik MVC
- gebruik van een logische package indeling

Eindpresentatie

Voor de eindpresentatie dienen jullie een afspraak te maken met je mentor. Jullie dienen dan de applicatie te presenteren/demonstreren aan de docent in een presentatie die zo'n 20-30 minuten zal duren. Jullie schrijven het verslag samen, maar ieder individu moet aanspreekbaar zijn op de inhoud.

Beoordeling

- Met het verslag kunnen maximaal 3 punten verkregen worden op het inhoudelijke aspect. Daarnaast vind er een aparte beoordeling plaats voor het onderdeel bedrijfscommunicatie door de docent BBC.
- Met de implementatie van de basisfunctionaliteit kunnen eveneens maximaal 3 punten verkregen worden.
- Door de extra functionaliteiten te implementeren kunnen maximaal 3 punten verkregen worden.
- De leden van de projectgroep voeren een peerreview uit waarbij een individuele bonus/malus wordt bepaald.
- De docent kan een individuele bonus/malus toekennen op basis van zijn waarnemingen of een afsluitend individueel gesprek.
- Het uiteindelijke resultaat dient minstens een 5.5 te zijn om te kunnen meewegen in het eindcijfer.

Herkansing

Indien het resultaat minstens een 4.5 is maar minder dan een 5.5, mag de groep uiterlijk in de eerste week van het opvolgende kwartaal een verbeteringsslag uitvoeren.