Programmeren 2
2010/2011 kw3
28 februari 2011

Eindopdracht

Rolit

Opdracht: Ontwikkel een gedistribueerd client/server programma waarmee het spel Rolit gespeeld kan worden, en beschrijf dit in een verslag.

Toelichting

Het doel van deze opdracht is om alle vaardigheden die bij Programmeren 1 en Programmeren 2 zijn aangeleerd te demonstreren, door het ontwikkelen van een client/server-applicatie waarmee het spelletje Rolit gespeeld kan worden.

Een speler kan met behulp van deze applicatie als client inloggen op de server. Als ook een tweede client heeft ingelogd, kan er een spelletje Rolit gespeeld worden. De applicatie dient te kunnen samenwerken met de applicaties van andere groepen (binnen jouw practicumgroep), zodat je tegen andere clients kunt spelen. Daarnaast moet je ook eenvoudige kunstmatige intelligentie aan de applicatie toevoegen, zodat het spel ook tegen een computerspeler gespeeld kan worden.

Dit document beschrijft het Rolit spel (par. 1), bespreekt de globale werking van de Rolit-applicatie (par. 2), definieert de functionele eisen aan de Rolit-applicatie die geïmplementeerd moet worden (par. 3) en bespreekt mogelijke uitbreidingen van deze applicatie (par. 4). Dit document geeft ook de eisen aan het programma (par. 5) en het verslag (par. 6), instructies voor de planning (par. 7), voor de peer review (par. 8) en voor het inleveren van de code en het verslag (par. 9), de regels van de boordeling (par. 10), een overzicht van alle belangrijke data (par. 11), en een opsomming van de beoordelingscriteria die door de docenten worden gehanteerd bij het nakijken van deze opdracht (par. 12).

Inhoudsopgave

1 Rolit: het spel

Rolit is een spel voor 2 tot 4 spelers.

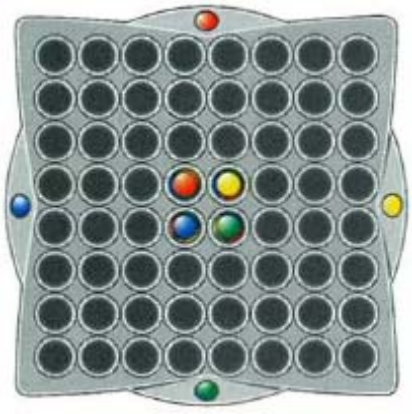
1.1 Materiaal en beginsituatie

Rolit wordt gespeeld met een bord van 8 x 8 = 64 posities en 64 Rolit-ballen (ballen met 4 kleuren die steeds gedraaid kunnen worden).

Iedere speler krijgt een kleur toegewezen die overeenkomt met een van de kleuren op de Rolit-ballen (rood, groen, blauw en geel). De toewijzing van kleuren gebeurt als volgt:

- Bij 2 spelers: één speler krijgt rood toegewezen en de andere groen.
- Bij 3 spelers: één speler krijgt rood toegewezen, de tweede geel en de derde groen.
- Bij 4 spelers: één speler krijgt rood toegewezen, de tweede geel, de derde groen en de vierde blauw.

Om het spel te beginnen worden de ballen op het bord gezet zoals aangegeven in Figuur 1. Deze beginsituatie is altijd hetzelfde, zowel voor 2, 3 als 4 spelers.



Figuur 1: Beginsituatie Rolit.

1.2 Doel van het spel

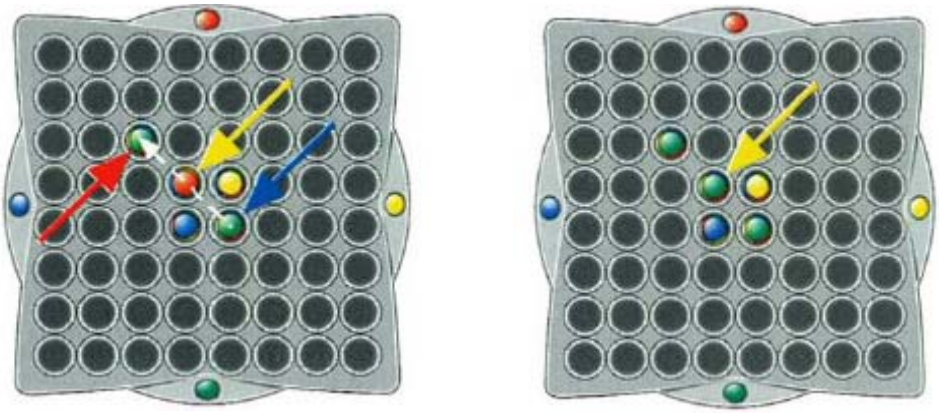
Doel van het spel is het blokkeren en veroveren van de Rolit-ballen van de tegenspelers door deze ballen naar je eigen kleur te rollen. De winnaar is diegene wiens kleur op het spelbord aan het eind het meest voorkomt.

1.3 Spelverloop

De speelvolgorde (wanneer is een speler aan de beurt) wordt bepaald via loting.

Als een speler aan de beurt is, plaatst hij een bal op een vrije positie naast een willekeurige Rolit-bal, en probeert zo ballen van de tegenstanders te blokkeren en te veroveren.

Figuur 2 laat zien hoe dat bijvoorbeeld vanuit de beginsituatie kan gebeuren. Er wordt een Rolit-bal op het speelbord gelegd, met de kleur van de speler 'naar boven', zodat deze een bal van een andere kleur blokkeert (rode pijl in Figuur 2(a)). Nu mag de geblokkeerde bal naar de kleur van de veroveraar gedraaid worden (gele pijl in Figuur 2(b)).



(a) (b)

Figuur 2. Veroveren van een kleur na de eerste zet.

De andere spelers plaatsen beurtelings ook steeds een bal op het bord en proberen op dezelfde manier één of meerdere Rolit-ballen van de tegenspelers te blokkeren en dan naar hun eigen kleur te rollen. Als je kleur verdwijnt of je niet kunt blokkeren, kun je een Rolit-bal naast elke willekeurige andere Rolit-bal op een vrije positie plaatsen.

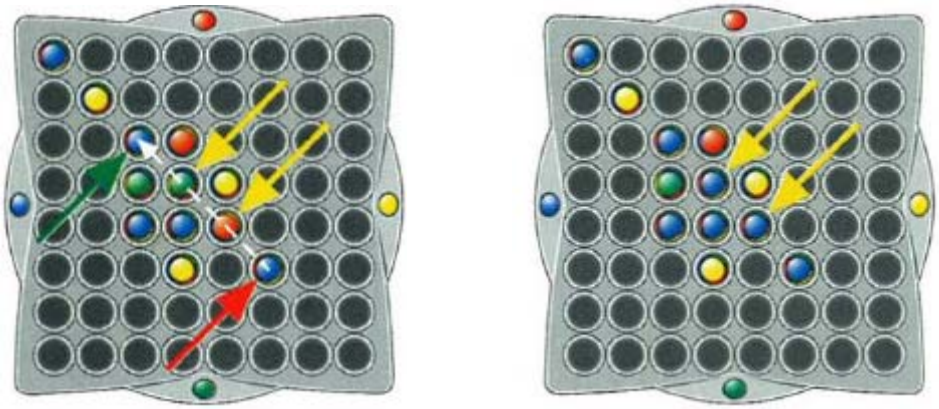
Blokkeren is uitstuitend mogelijk tussen de zojuist geplaatste bal en de volgende reeds liggende bal van dezelfde kleur, en wel in een rechte lijn (blauwe pijl in Figuur 2(a)). De geblokkeerde bal (gele pijl in Figuur 2(a)) wordt nu veroverd door deze naar de kleur van de veroveraar te rollen (gele pijl in Figuur 2(b)).

Er zijn een aantal belangrijke spelregels:

1. Een bal op het spelbord wordt altijd geplaatst naast een reeds liggende bal.
2. Een speler mag slechts één bal per beurt op het spelbord plaatsen.
3. Als je kunt blokkeren en veroveren, dan ben je verplicht dit ook te doen.

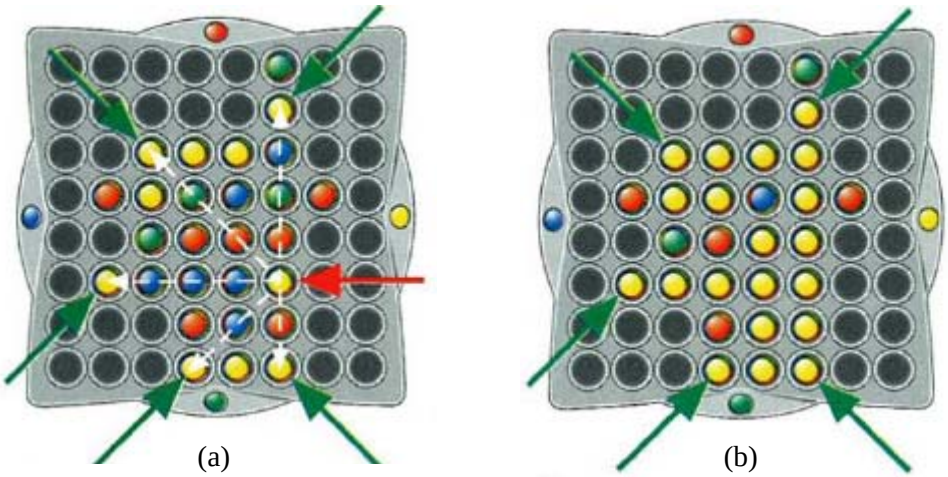
Figuur 3 laat de begin- en eindsituatie van een zet zien. Blokkeren is uitsluitend mogelijk tussen de zojuist geplaatste blauwe bal (rode pijl in Figuur 3(a)) en de eerstvolgende blauwe bal (groene pijl in Figuur 3(a)). De groene pijl markeert dus de grens van de blokkade. De geblokkeerde ballen (gele pijlen in Figuur 3(a)) worden nu veroverd en naar de kleur van de veroveraar gerold (gele pijlen in Figuur 3(b)).

Figuur 4 laat de begin- en eindsituatie van een andere zet zien. De Rolit-ballen kunnen geblokkeerd en veroverd worden in alle richtingen (horizontaal, verticaal en diagonaal), maar uitsluitend in rechte lijnen vanuit de nieuwe geplaatste bal. In Figuur 4(a) markeert de rode pijl de geplaatste gele bal, en de groene pijlen markeren de grenzen van de blokkades. In Figuur 4(b) zien we het resultaat.



(a) (b)

Figuur 3: Begin- en eindsituatie van een zet.



(a) (b)

Figuur 4: Begin- en eindsituatie van een andere zet.

1.4 Einde van het spel

Het spel is afgelopen wanneer alle Rolit-ballen op het spelbord geplaatst zijn. De winnaar is diegene wiens kleur het meest voorkomt op het spelbord.

Waarschuwing

Het is misschien mogelijk om een implementatie van Rolit te vinden op het Internet, maar het is niet de bedoeling dat je zo'n oplossing kopieert ('het gaat om het spel, niet om de knikkers!'). Bovendien gebruiken de beschikbare oplossingen waarschijnlijk een ander protocol, zodat het aanpassen meer tijd zal kosten dan zelf maken.

In ieder geval moet je daarom op elk moment in staat zijn om jouw (eigen) oplossing uit te leggen aan de docenten. Mocht er worden geconstateerd dat een oplossing ergens vandaan is gekopieerd, dan wordt de desbetreffende groep uitgesloten van dit onderdeel en wordt dit voorgelegd aan de examencommissie.

2 Globale werking Rolit-applicatie

Na het opstarten van een client (GUI), kan een gebruiker van de client het internetadres en het poortnummer van de server opgeven en zijn eigen naam invoeren. Daarna kan hij contact zoeken met de server. De client blijft wachten totdat de server vertelt dat er een andere client is ingelogd die een spel wil spelen. Als een tweede client zich heeft aangemeld, kan het spel beginnen of kan er op meer

spelers gewacht worden tot een maximum van vier. Als het spel mag beginnen kan de server willekeurig de kleuren toewijzen aan de spelers of de speler laten kiezen in volgorde van aanmelding. Als het spel begonnen is, blijft de server wachten op nieuwe clients die Rolit willen spelen. Er kunnen derhalve meerdere spelletjes tegelijk door één Rolit server verzorgd worden.

Het spel zelf zou vervolgens als volgt kunnen verlopen. De speler die aan de beurt is, kiest een lege positie om zijn bal te plaatsen, met inachtneming van de spelregels. De client controleert deze 'zet' en als die geldig is wordt de zet naar de server gestuurd.

De server controleert ook de geldigheid van de zet en als de zet geldig is wordt deze naar alle clients gezonden die vervolgens het speelbord bijwerken. De beurt gaat vervolgens naar een andere speler die op zijn beurt ook een geldige zet moet maken. Het spel wordt dan verder gespeeld volgens de spelregels totdat het bord vol is. Hierna wordt de winnaar bepaald.

Opmerking

Het zou heel goed kunnen dat in jouw practicumgroep een ander protocol wordt afgesproken, waarbij de verantwoordelijkheden van de client en server anders liggen dan wat hierboven wordt beschreven.

2.1 Communicatie

Het is de bedoeling dat jouw client-applicatie met de server van andere groepjes (binnen één practicumgroep) kan communiceren en vice versa. Dit betekent dat alle koppels binnen een practicumgroep hetzelfde *protocol* moeten gebruiken voor de client/server-communicatie. Het protocol schrijft voor welke data tussen de client en server wordt verstuurd en in welke volgorde.

Er kan bijvoorbeeld afgesproken worden dat de client en de server communiceren met behulp van berichten van de klasse `String` gebruikmakend van `Reader`- en `Writer`-objecten[1]. In dit geval zou een bericht de volgende vorm kunnen hebben:

```
<commando> <arg1> <arg2> <arg3>
```

waarbij commando's en argumenten worden gescheiden door spaties (In commando's en argumenten kunnen dus geen spaties voorkomen[2].) Een voorbeeld van een commando met één argument is

```
join Doornroosje
```

waarmee een client bij een server aangeeft dat een gebruiker met de naam `Doornroosje` wil inloggen om een spel te spelen.

De vraag is nu welke commando's (en strings) er nodig zijn zodat de communicatie tussen de client en de server goed verloopt. Het werkcollege van week 5 is bedoeld om hierover te overleggen en een bepaald communicatieprotocol vast te leggen. Tijdens dit werkcollege dient dus het formaat van de commando's en de argumenten te worden vastgesteld. Er dient overeenstemming te worden bereikt over de *minimale* verzameling van berichten die de client en server moeten begrijpen. Het is natuurlijk toegestaan om voor je eigen client/server-applicatie het protocol uit te breiden met extra berichten die nodig zijn voor extra functionaliteit van de applicatie.

Het protocol moet op een duidelijke manier beschreven staan, zodat alle koppels client en server implementaties kunnen maken die aan het protocol voldoen. Alle commando's (d.w.z. de strings in het voorbeeld hierboven) kunnen in een aparte Java klasse opgenomen worden en deze klasse kan op de Blackboard site van het vak ter beschikking gesteld worden aan alle leden van de practicumgroep.

3 Definitie van functionele eisen

Hieronder volgt een lijst van functionele eisen voor de Rolit-applicatie.

Eisen aan de *server*:

[1] Een alternatief is om deze informatie in 'binaire vorm' te coderen, zoals behandeld in hoorcollege 2.

[2] Dit betekent dat als er een argument nodig is dat wel spaties bevat, deze spaties vóór het versturen van het bericht vervangen moeten worden door andere karakters, die na het ontvangen weer omgezet moeten worden in spaties.

1. Bij het opstarten van de server moet een poortnummer worden opgegeven waarnaar de server zal luisteren. Als de server een GUI heeft, dan mag dit poortnummer ook op de GUI ingevuld worden.
2. Als het opgegeven poortnummer in gebruik is, wordt een passende foutmelding gegeven, waarna een ander nummer kan worden geprobeerd.
3. Een server moet verschillende Rolit-spelletjes tegelijkertijd kunnen verzorgen tussen verschillende clients.
4. Als de server een GUI heeft, dan dienen alle communicatie-berichten in een `(J)TextArea` getoond te worden en de GUI dient schaalbaar te zijn. Als de server een command-line (Terminal) applicatie is, dan moeten alle communicatie-berichten naar `System.out` geschreven worden.
5. Zowel de server als de client dienen zich te houden aan het protocol zoals dat gedefinieerd is in het werkcollege van week 5, d.w.z. de server dient met alle andere clients van jouw practicumgroep te kunnen samenwerken, en de client dient met alle andere servers van jouw practicumgroep te kunnen samenwerken.

Eisen aan de *client*:

1. De client GUI is schaalbaar en biedt de gebruiker verschillende opties (bijv. poortnummer, IP-adres) om een spel bij een server aan te vragen.
2. De client moet een menselijke gebruiker kunnen laten spelen of zich als computerspeler met (enige) kunstmatige intelligentie gedragen.
3. Een (menselijke) gebruiker van de client GUI moet een spel kunnen beëindigen.
4. Bij het spelen van Rolit hebben beide clients en de server altijd dezelfde kijk op de toestand van het spel.
5. De bedenktijd van een computerspeler (en derhalve de kracht van de kunstmatige intelligentie) moet op de client GUI instelbaar zijn.
6. De client biedt een hint-functie aan, waarmee een menselijke speler een voorstel vraagt voor een zet die gemaakt zou worden door de computerstrategie. De zet mag alleen voorgesteld worden; de menselijke speler moet zelf de voorgestelde zet daarna spelen of kiezen voor een andere zet.
7. Na afloop van een spel moet een speler een nieuw Rolit-spel kunnen beginnen.
8. Als een speler het spel voortijdig beëindigt, of hij/zij sluit zijn GUI af, of de client-applicatie crashed, moet de andere speler daarvan op de hoogte worden gesteld. De andere speler zou zich weer opnieuw kunnen aanmelden bij de server om een spelletje Rolit te spelen.
9. Een server kan ten allen tijde onverwachts de verbinding verbreken. Dit moet door de verbonden clients op een nette manier afgehandeld worden.

4 Uitbreidingen voor bonus punten

Wellicht dat je tijd over hebt om de Rolit-applicatie verder uit te breiden. Hoe meer functionaliteit jouw Rolit-applicatie biedt, hoe hoger het cijfer van de practicumopdracht zou kunnen zijn. Je zou de volgende functies kunnen overwegen.

4.1 Chat-functie

Een Rolit-applicatie ontwikkeld volgens de eisen die wij tot nu toe hebben gesteld kan gebruikers alleen een spelletje Rolit laten spelen. Het is natuurlijk leuker als spelers ook met elkaar kunnen praten tijdens het spel. Een mogelijke uitbreiding is dus een veld aan de client-GUI toevoegen waarin de gebruiker teksten kan intypen. Na het indrukken van de returnknop dient het bericht naar de server gestuurd te worden, die het vervolgens naar alle aangesloten clients stuurt. Het bericht dient vervolgens op alle client-GUIs getoond te worden.

4.2 Challenge-functie

Tot dusver werden clients in volgorde van aanmelden aan elkaar gekoppeld voor het spelen van een spelletje Rolit. Het is natuurlijk fraaier als een client zou kunnen kiezen uit de aangemelde clients om

een spelletje Rolit mee te spelen. Om deze uitbreiding te realiseren dienen tenminste de volgende veranderingen te worden aangebracht:

- Een client dient te weten welke andere clients ingelogd zijn.
- Een client moet kunnen kiezen met welke client hij een spelletje Rolit zou willen spelen.
- Een client moet een spelletje kunnen weigeren.

De uitbreidingen worden alleen beoordeeld als de applicatie aan de basis functionele eisen voldoet (d.w.z., de basisfunctionaliteit op de juiste manier werkt).

5 Vereiste programma-onderdelen

Het opgeleverde programma dient de volgende onderdelen te bevatten:

1. Een README file met daarin aanwijzingen voor de installatie en het opstarten van het programma, zoals de voor executie noodzakelijke directories en files en de manier waarop e.e.a. geïnstalleerd en aangeroepen dient te worden. Bij het lezen van deze file moet een gebruiker in staat zijn het programma foutloos te compileren en op te starten.
   Indien hieraan niet voldaan is, komt het programma niet voor beoordeling in aanmerking; niet-compileerbare programma's worden niet geaccepteerd.
2. De Java code van alle zelfgedefinieerde klassen, in één enkele directory-hiërarchie. De code dient aan de volgende eisen te voldoen:
   - Foutvrij te compileren (onder Java 1.6); *
   - Zinvol gebruik van packages en toegankelijkheden;
   - Begrijpelijke opmaak en naamgeving, volgens Java-conventies;
   - Specificatie (in Javadoc) van klassen en methoden, inclusief pre- en postcondities en klasseinvarianten, en een toelichting bij elke klasse die niet in het verslag is opgenomen; *
   - Commentaar bij de implementatie van grotere/complexere methoden, waar nodig met lusinvarianten.

   *De met een sterretje * gemarkeerde eisen zijn noodzakelijk om voor de eindopdracht 50 punten of meer te krijgen.*
3. Documentatie (door Javadoc geproduceerd in HTML) van alle zelfgedefinieerde klassen, in een eigen directory-hiërarchie (dus niet gecombineerd met de Java-files).
4. Bytecode van eventuele gebruikte voorgedefinieerde klassen, voor zover het geen zelfgeprogrammeerde of standaard Java klassen betreft.
5. Alle gebruikte testklassen en testruns.

Zorg ervoor dat het systeem zodanig is opgeleverd dat de beoordelaar na het lezen van de bijgeleverde README file het programma kan compileren en draaien. *Het mag dus niet nodig zijn iets in de source-code te veranderen!* Typische oorzaken van de gevallen waarin dit fout gaat zijn namen en paden van files of andere URLs, zoals van hostmachines van servers. *Test dit alvorens het product in te leveren.*

Het is aan te bevelen om de ‘Export’ functionaliteit van Eclipse te gebruiken om het hele Eclipse project naar de .zip file te kopiëren.

6 Vereiste verslagonderdelen

Het *verslag* moet in ieder geval de volgende onderdelen bevatten:

1. Toelichting op het ontwerp;
2. Toelichting per klasse;
3. Testverslag.

Een nadere uitleg van deze verslagonderdelen volgt hieronder. Hierbij is telkens aangegeven welke functie het betreffende verslagonderdeel heeft, en welke doelgroep je voor ogen moet hebben bij het schrijven ervan. Deze onderdelen moeten bovendien ingebed zijn in een samenhangend geheel, dat zich als één verslag laat lezen.

De source-code hoeft niet in het verslag opgenomen te worden, net als de met Javadoc gegenereerde HTML files.

Tenslotte: vermeit taalvoute en tikfoutn. Se staan so dullig.

6.1 Toelichting op het ontwerp

Het ontwerp van het programma is de globale structuur van het systeem, beschreven in termen van de klassen met hun samenhang. Het verslag hiervan moet de volgende elementen bevatten:

1. Becommentarieerde klassediagrammen, bijvoorbeeld per package en globaal, zodanig weergegeven dat hieruit de structuur goed op te maken is. Hiervoor dient je met behulp van Together een opmaak van de diagrammen te kiezen die overzichtelijk is, bijvoorbeeld door het positioneren van klassen en het weglaten van onbelangrijke delen. Schuw daarbij ook het gebruik van extra labels en notities niet!
2. Een systematische opgave van welke delen van de functionaliteit (zoals geformuleerd in de definitie van functionele eisen) door welke klassen worden geïmplementeerd. Als je aan bepaalde stukken van de gevraagde functionaliteit niet toegekomen bent, is dit de plaats om dat aan te geven;
3. Het gebruik van *observer*- en *model-view-controller*-patronen;
4. Formaten voor file-opslag en communicatieprotocollen (tussen parallel lopende programma-onderdelen). Als er protocollen klassikaal zijn besproken en gedefinieerd kun je volstaan met een verwijzing daarnaar.

Doelgroep

Dit verslagonderdeel is bedoeld voor het “onderhoudspersoneel” dat het programma niet geschreven heeft, maar het bij een latere gelegenheid moet verbeteren of uitbreiden, en daarvoor moet kunnen begrijpen hoe het ontwerp van uw programma in elkaar steekt.

6.2 Toelichting per klasse

Bij elke klasse in het systeem dient een toelichting gegeven te worden, die de volgende elementen moet bevatten:

1. De rol die de klasse in het systeem speelt;
2. De verantwoordelijkheden die de klasse heeft;
3. De andere klassen waarmee deze klassen samenwerkt om haar verantwoordelijkheden te realiseren;
4. Eventuele bijzonderheden in het contract van de klasse;
5. Eventuele maatregelen die getroffen zijn om aan de precondities in het contract van de server-klassen (van deze klasse) te voldoen.

Doelgroep

Dit verslagonderdeel is bedoeld voor het “onderhoudspersoneel” dat bij een latere gelegenheid een of meerdere klassen moet verbeteren of uitbreiden, en daarvoor doel en werking van de klasse moet kunnen begrijpen.

6.3 Testverslag

Het grondig testen van het systeem maakt onlosmakelijk deel uit van de ontwikkeling ervan. Hierin zijn tenminste twee manieren van testen te onderscheiden, die allebei in het verslag moeten worden opgenomen.

In het testverslag is het van belang dat je eventuele fouten die je zelf bij het testen ontdekt hebt en uit tijdsgebrek in het programma zijn blijven staan ook aangeeft.

Doelgroep

Dit verslagonderdeel is bedoeld voor het “onderhoudspersoneel” dat bij een latere gelegenheid één of meerdere klassen moet verbeteren of uitbreiden, en daarna moet kunnen uitproberen of in ieder geval de reeds gedane tests nog steeds met succes doorstaan worden.

Unit-testen

Hierbij wordt de functionaliteit van de klassen van het systeem zoveel mogelijk afzonderlijk getest. Er bestaan verschillende technieken, met een wisselende mate van grondigheid:

1. De meest grondige en uitgebreide manier van testen gebeurt met behulp van speciaal daarvoor geschreven testklassen, zoals u met name bij Programmeren 1 hebt geleerd.
2. Wanneer het programma via het netwerk communiceert, kun je delen van het systeem (bijvoorbeeld een server) testen door met behulp van `telnet` andere systeemdelen (een of meerdere clients) handmatig te `simuleren'. In dit geval test je niet een enkele klasse afzonderlijk, maar meerdere tegelijk.
3. Een meer oppervlakkige manier van testen is om voor testdoeleinden een `main` aan een klasse toe te voegen die een of meerdere objecten van de klasse maakt en er enige methoden op aanroept.
4. Het testen van een GUI-klasse bestaat vaak uitsluitend uit het visueel beoordelen van de lay-out.

Het unit-testverslag dient voor elke klasse de volgende zaken aan te geven:

1. Hoe de klasse getest is: afzonderlijk, tezamen met andere klassen of in het geheel niet;
2. Welke van bovenstaande testtechnieken daarbij toegepast zijn;
3. Eventuele testprogramma's die hiervoor ontwikkeld zijn;
4. Testdoorlopen met verwachte uitkomsten.

Schrijf dit zó op dat de lezer de tests eventueel zelf kan herhalen. Testprogramma's en -doorlopen moeten bij het opgeleverde programma zijn gevoegd.

Systeemtesten

Bij het systeemtesten wordt het opgeleverde, werkende programma *als geheel* uitgeprobeerd. Hierbij zal de definitie van eisen als uitgangspunt genomen moeten worden: het programma dient in ieder geval de daarin beschreven functionaliteit correct te implementeren. Elk van de items in de definitie van eisen kan afzonderlijk worden getest door één of meerdere zogenaamde testdoorlopen, waarin verschillende scenario's (met correct dan wel foutief gebruik) worden doorgespeeld.

Ook de uitgevoerde systeemtests moeten in het verslag beschreven worden. Beschrijf opnieuw welke aspecten van het systeem je hebt getest, en op welke manier, onder verwijzing naar de definitie van eisen. Deze beschrijving dient te bestaan uit:

1. De geteste functionaliteit, onder verwijzing naar de definitie van eisen;
2. De gebruikte testdoorloop, beschreven op een manier die herhaling door de lezer toelaat;
3. De bedoelde (gewenste) reactie van het systeem;
4. De werkelijke reactie, als die verschilt van de gewenste.

7 Planning

Om de opdracht op een goede manier te kunnen maken, wordt je gevraagd om zelf een planning te maken en die met de practicumassistent te bespreken. Voor de planning kun je het formulier in Bijlage A gebruiken, maar je mag ook een eigen formulier gebruiken. Op donderdag in week 5 (10 maart 2011) moet je je planning met de practicumassistent bespreken. Hij zal je eventueel suggesties geven hoe je je planning zou kunnen aanpassen, als hij denkt dat deze niet reëel is. Het uiteindelijke doel van de eindopdracht is om een werkend programma en verslag op te leveren. Dat heeft de practicumassistent zelf ook gedaan; neem daarom zijn suggesties serieus.

Vervolgens bespreek je één keer per week kort met de practicumassistent de voortgang van je project aan de hand van de planning. De practicumassistent zal deze planninggesprekken ook aftekenen; zij zijn noodzakelijk voor het behalen van een voldoende voor de eindopdracht.

7.1 Werkwijze

Hieronder volgen een aantal tips voor het maken en bijhouden van de planning:

– Verdeel de (grote) werkopdracht in kleinere opdrachten. Op deze manier wordt de grote opdracht concreter, in de vorm van kleinere opdrachten waaraan je meteen kunt beginnen.

Voor deze opdracht kun je bijvoorbeeld een aantal onderdelen van het programma onderscheiden: de implementatie van het eigenlijke spel (speelbord, zetten, enzovoorts), de grafische user interface, het spelen over het netwerk en de kunstmatige intelligentie.

- Je moet uiteindelijk zowel een programma als een verslag inleveren. Houd hier in je planning ook rekening mee, d.w.z. geef op elk moment aan wat je aan het verslag gaat doen en wat je aan het programma gaat doen. Natuurlijk zitten hier ook verschillende afhankelijkheden tussen; formuleer deze afhankelijkheden expliciet.
- Formuleer voor de opdrachten (haalbare) doelen en probeer deze SMART te formuleren (zie bijv. http://www.carrieretijger.nl/functioneren/management/leidinggeven/doelen-stellen/smart voor een korte uitleg hierover). Een SMART doel is duidelijk geformuleerd, realistisch en meetbaar, d.w.z. je kunt precies aangeven wanneer het doel behaald is.
- Maak een lijst van acties (wat moet je doen om de doelen te halen?).
- Stel prioriteiten (wat moet je het eerst doen en wat is het meest belangrijk).
- Maak een overzicht van de beschikbare tijd.
- Stel nu een actieplan op dat aangeeft wie wanneer welke actie gaat uitvoeren (en wat het verwachte resultaat is van de actie).
- Ken je eigen werkritme (bioritme)
- Houd rekening met het onverwachte (plan wat extra tijd)

7.2 Reflecteren op je eigen planning

In het standaardformulier is een kolom waarin je ook de daadwerkelijk besteedde tijd kunt aangeven. De practicumassistent zal je hier niet op beoordelen; hij zal alleen kijken of je de gestelde doelen ook gehaald hebt en als er problemen zijn overleggen hoe je weer “op schema” kunt raken. We raden je echter aan om voor jezelf deze kolom wel in te vullen, omdat dit je later kan helpen bij het maken van andere planningen en bij het reflecteren op je eigen prestaties.

Tenslotte nog een tip hoe je je zo goed mogelijk aan je eigen planning kunt houden: laat je niet afleiden door allerlei tijdsverspillers, zoals slecht toegankelijke informatie, geen prioriteiten stellen, en de planning als wet zien en niet willen aanpassen als het nodig blijkt te zijn.

8 Peer reviews

Halverwege de eindopdracht moet je een voorlopige versie van je verslag, met daarin onderdeel 1 en 2 inleveren, om zo feedback te krijgen van je medestudenten. De indeling voor de peer reviews wordt op Blackboard bekend gemaakt. De studenten regelen onderling dat het conceptverslag op vrijdag 25 maart naar de medestudenten voor peer review wordt verstuurd. De peer reviews worden op de dinsdag in week 8 (29 maart) afgetekend door de practicumassistent, tijdens het practicumuur op dinsdag. Het inleveren van een peer review is een voorwaarde voor het behalen van een cijfer voor de eindopdracht. d.w.z., groepen die de peer review niet hebben afgetekend worden uitgesloten van dit practicumonderdeel.

Geef constructief commentaar waar je medestudenten mee verder kunnen. Probeer alle genoemde onderdelen zo goed mogelijk te beoordelen en geef waar nodig een toelichting, bijvoorbeeld aan de hand van een voorbeeld. Feedback kan ook mondeling worden toegelicht; mondeling toelichting van de peer review is niet verplicht maar wordt sterk aanbevolen. Bedenk dat je medestudenten dit ook voor jou doen, dus geef serieus commentaar en raffel dit niet af. Bovendien, als je ziet hoe anderen de dingen opzetten heb je er misschien ook wat aan voor je eigen verslag.

Het formulier voor het geven van peer reviews is in Bijlage B van dit document te vinden.

9 Inleveren

Het verslag wordt ingeleverd binnen de gestelde termijn bij de hoofddocent. Het verslag dient dubbelzijdig afgedrukt op A4 papier en zonder kaft of ringband ingeleverd te worden; maak de bladzijden gewoon aan elkaar vast met een nietje en lever het verlag zo in. Het is de inhoud die beoordeeld wordt.

De code wordt binnen de gestelde termijn via Blackboard ingeleverd als een .zip file met alle code die nodig is om het programma te compileren en executeren (zie ook par. 5).

Inleveren per email is niet toegestaan. Wanneer het ingeleverde op wezenlijke punten afwijkt van het hier gevraagde komt het niet voor beoordeling in aanmerking.

10 Beoordeling

Het opgeleverde programma en het verslag worden apart beoordeeld met een cijfer tussen 0 en 100, en het eindcijfer van de eindopdracht is het gemiddelde cijfer van deze twee onderdelen. De basis functionele eisen, vereiste programma-onderdelen en vereiste verslagonderdelen zijn noodzakelijke maar geen voldoende voorwaarden om voor de eindopdracht een eindcijfer te krijgen dat gelijk of hoger is dan 50 punten. Dat betekent dus dat als één van de genoemde verslagonderdelen duidelijk onder de maat is het eindcijfer lager zal zijn dan 50 punten.

Protocol

In week 5 wordt het protocol afgesproken. Dit protocol wordt verder gedocumenteerd en bijgehouden door één student uit de practicumgroep. Als beloning krijgt deze student 10 bonuspunten, mits deze taak naar behoren is uitgevoerd. Deelname aan dit werkcollege is verplicht.

Toernooi

In de laatste week van het practicum wordt er een toernooi georganiseerd waarin de Rolit-clients (kunstmatige intelligentie) tegen elkaar spelen om te bepalen wie de winnaar is binnen elke practicumgroep (INF1, INF2, INF3 en BIT). De winnaar van het toernooi krijgt 10 bonuspunten en de tweede krijgt 5 bonuspunten (opgeteld bij het eindcijfer). Als een Rolit-implementatie niet werkt tijdens het toernooi, krijgen de studenten 10 punten aftrek op het cijfer van de eindopdracht. De studenten hebben tot het eind van de practicumsessie de tijd om een werkende implementatie te laten zien.

Bonus- en aftrekregeling

Samenvattend isde bonus- en aftrekregeling dus als volgt:

1. Voor elke uitbreiding (chat en challenge): 5 bonuspunten.
2. Voor het bijhouden van het protocol: 10 bonuspunten.
3. Voor de winnaar van het toernooi: 10 bonuspunten.
4. Voor de tweede van het toernooi: 5 bonuspunten.
5. Voor een niet werkende implementatie tijdens het toernooi: 10 aftrekpunten.

Wanneer de docenten dit nodig achten, kunnen ze een mondelinge toelichting vragen op het ingeleverde programma of het verslag.

11 Belangrijke data

maandag 7 maart 2011	hoorcollege met uitleg over eindopdracht
woensdag 9 maart 2011	werkcollege waarin protocol wordt vastgesteld
donderdag 10 maart 2011	planning bespreken met practicumassistent (met wekelijkse update)
vrijdag 25 maart 2011	verslag onderdeel 1 en 2 inleveren voor peer feedback
dinsdag 29 maart 2011	peer feedback inleveren en aan mede-studenten geven tijdens practicumuren
donderdag 31 maart 2011	toernooi tijdens practicumuren
donderdag 7 april 2011	deadline voor inleveren definitieve code en verslag

12 Beoordelingscriteria

Hieronder geven wij een opsomming van de beoordelingscriteria die de docenten zullen hanteren voor het nakijken van deze opdracht. Gebruik zelf deze opsomming om te controleren of je eigen code en verslag aan de beoordelingscriteria van de docenten voldoen.

12.1 Code

De code van de Rolit-applicatie wordt beoordeeld aan de hand van de volgende criteria:

1. Het programma bevat de vereiste onderdelen (zie functionele eisen hierboven).
2. Er is een README file met aanwijzingen voor installatie en opstarten.
3. Het programma compileert zonder fouten.
4. Het programma executeert zonder fouten.
5. Het programma is van Javadoc commentaar voorzien, inclusief pre- en postcondities en klasseinvarianten.
6. Naast de Javadoc voor alle methoden, is de implementatie van grotere en complexere methoden van commentaar voorzien, inclusief lus-invarianten.
7. Het programma is op een begrijpelijke manier opgemaakt.
8. Er wordt zinvol gebruik gemaakt van packages en toegankelijkheden.
9. Het programma bevat een grafische user interface.
10. Bytecode van voorgedefinieerde, externe, niet-standaard Java, klassen is meegeleverd.
11. Alle gebruikte testklassen en testruns zijn meegeleverd.

De volgende criteria hebben betrekking tot het ontwerp van de Rolit-applicatie:

1. Het algemene ontwerp is logisch van opzet.
2. De implementatie komt overeen met het ontwerp in het verslag.
3. Het Model-View-Controller patroon komt goed tot uitdrukking.
4. Het programma is logisch opgedeeld in klassen.
5. Alle klassen zijn gedetailleerd gedocumenteerd.

De volgende criteria hebben betrekking tot de programmeerstijl:

1. Namen van klassen, variabelen en methoden zijn logisch gekozen.
2. Code is efficiënt en netjes geïmplementeerd.
3. Het programma is goed onderhoudbaar (gebruik van constanten, variabelenamen etc.).
4. Het communicatieprotocol zoals afgesproken in de groep is op een juiste manier geïmplementeerd.
5. Het exceptiemechanisme wordt op een juiste manier gebruikt.
6. De grafische user interface ziet er verzorgd uit. De interface en de onderdelen daarvan zijn schaalbaar.
7. Concurrency constructs zijn op een juiste manier gebruikt.
8. De computerspeler gebruikt een intelligente strategie.

De volgende criteria gelden voor het behalen van bonus punten:

1. De chat-functionaliteit is correct geïmplementeerd en gedocumenteerd (5 punt extra voor de eindopdracht).
2. De challenge-functionaliteit is correct geïmplementeerd en gedocumenteerd (5 punt extra voor de eindopdracht).

12.2 Verslag

Het verslag van de Rolit-applicatie wordt beoordeeld aan de hand van de volgende criteria:

1. Het verslag bevat de vereiste onderdelen.
2. Het algemene ontwerp is duidelijk beschreven
3. De klassediagrammen zijn op een begrijpelijke manier beschreven
4. De gemaakte keuzen in het algemene ontwerp zijn duidelijk toegelicht
5. Het algemene ontwerp is voor de juiste doelgroep (software maintainer) beschreven
6. Er wordt een overzicht gegeven welke functionaliteit door welke klassen geïmplementeerd wordt
7. Er wordt beschreven hoe van observer en model-view-controller patronen gebruik wordt gemaakt
8. Formaten voor file-opslag en communicatieprotocollen zijn beschreven
9. De rol en verantwoordelijkheden van de verschillende klassen zijn duidelijk beschreven
10. De gemaakte keuzen in de klasse-implementaties zijn duidelijk toegelicht
11. De klasse-implementaties zijn voor de juiste doelgroep (software maintainer) beschreven

12. De connecties tussen klassen zijn duidelijk beschreven
13. Eventuele bijzonderheden in het “contract” van de klasse zijn duidelijk beschreven.
14. De test-activiteiten zijn duidelijk beschreven
15. Alle klassen van het systeem worden door middel van unit tests getest
16. De test-activiteiten omvatten het gehele programma
17. De testresultaten zijn geanalyseerd
18. Het testverslag is voor de juiste doelgroep (software maintainer) geschreven

De volgende criteria hebben betrekking tot de structuur van het verslag:

1. De lezer kan snel en trefzeker zien waar de tekst over gaat.
2. De tekst is logisch geordend.
3. De benodigde informatie kan makkelijk gevonden worden.

De volgende criteria hebben betrekking tot taalgebruik in het verslag:

1. Er is sprake van begrijpelijk taalgebruik: woordkeus en zinsbouw zijn afgestemd op de beoogde lezers.
2. De spelling is correct en consistent.
3. De zinnen in de tekst zijn goed geconstrueerd.

Bijlage A: Planning formulier

Week	**Verplichte Onderdelen**	**Doelen** - **Programma** - **Verslag**	**Acties (wie)**	**Prioriteiten**	**Beschikbare tijd**	**Tijd die je wilt besteden**	**Doel gehaald?**	**Besteedde tijd**
5 (6/3/2011-12/3/2011)	**7/3/2011**: HC met uitleg over opdracht **10/3/2011**: planning bespreken met practicumassistent	- Programma: - Verslag:						
6 (13/3/2011-19/3/2011)		- Programma: - Verslag:						
7 (20/3/2011-26/3/2011)	**25/3/2011**: verslag onderdeel 1 + 2 inleveren voor peer feedback	- Programma: - Verslag:						
8 (27/3/2011-2/4/2011)	**29/3/2011**: peer feedback op verslag medestudenten geven **31/3/2011**: toernooi	- Programma: - Verslag:						
9 (3/4/2011 – 9/4/2011)	**7/4/2011**: deadline	- Programma: - Verslag:						

Bijlage B: Peer review formulier

Naam (Studentnummer):
Naam (Studentnummer):

Dit formulier wordt gebruikt om feedback te geven op de voorlopige versie van het verslag van medestudenten binnen de eigen practicumgroep voor de eindopdracht van Programmeren 2. Geef constructief commentaar, waar je medestudenten mee verder kunnen. Probeer alle genoemde onderdelen zo goed mogelijk te beoordelen en geef waar nodig een toelichting, bijvoorbeeld aan de hand van een voorbeeld. De feedback kan ook verder mondeling worden toegelicht.

Bedenk dat je medestudenten dit ook voor jou doen, dus geef serieus commentaar en raffel dit niet af. En misschien heb je er ook wat aan voor je eigen verslag, als je ziet hoe anderen de dingen opzetten.

Eisen	onvol-doende	matig	vol-doende	goed
Het verslag bevat de vereiste onderdelen				
Evt. Toelichting				

Inhoud	onvol-doende	matig	vol-doende	goed
Het algemene ontwerp is duidelijk beschreven				
De klassediagrammen zijn op een begrijpelijke manier beschreven				
De gemaakte keuzen in het algemene ontwerp zijn duidelijk toegelicht				
Het algemene ontwerp is voor de juiste doelgroep (software maintainer) beschreven				
Er wordt een overzicht gegeven welke functionaliteit door welke klassen geïmplementeerd wordt				
Er wordt beschreven hoe van observer en model-view-controller patronen gebruik wordt gemaakt				

Inhoud (vervolg)	onvol-doende	matig	vol-doende	goed
Formaten voor file-opslag en communicatieprotocollen zijn beschreven				
De rol en verantwoordelijkheden van de verschillende klassen zijn duidelijk beschreven				
De gemaakte keuzen in de klasse-implementaties zijn duidelijk toegelicht				
De klasse-implementaties zijn voor de juiste doelgroep (software maintainer) beschreven				
De connecties tussen klassen zijn duidelijk beschreven				
Eventuele bijzonderheden in het “contract” van de klasse zijn duidelijk beschreven.				
Evt. Toelichting				

Structuur	onvol-doende	matig	vol-doende	Goed
De lezer kan snel en trefzeker zien waar de tekst over gaat				
De tekst is logisch geordend				
De benodigde informatie kan makkelijk gevonden worden				
Evt. Toelichting				

Taalgebruik	onvol-doende	matig	vol-doende	Goed
Er is sprake van begrijpelijk taalgebruik: woordkeus en zinsbouw zijn afgestemd op de beoogde lezers.				
De spelling is correct en consistent.				
De zinnen in de tekst zijn goed geconstrueerd.				
Evt. Toelichting				