### GSO3 Opdracht Unittesten en Specificeren

**Inleiding**

Met dit beroepsproduct kun je laten zien dat je de volgende vaardigheden beheerst:

* ik kan een gedegen unittest schrijven, dat betekent onder meer dat ik kritisch de vereiste grensgevallen vaststel en dat mijn testcode eenvoudig door een ander kan worden gecontroleerd;
* ik kan bij het programmeren van een klasse effectief gebruikmaken van een gegeven unittest;
* ik kan de methoden en constructoren van een klasse eenduidig en helder met javadoc specificeren;
* ik kan een gegeven specificatie van een klasse kritisch reviewen;
* ik kan een gegeven unittest kritisch reviewen.

**Werkwijze**

Gegeven is de aangeleverde code in de package fontys.time. Daarin zitten de twee interfaces ITime en ITimeSpan, de enum DayInWeek en de twee klassen ITime en ITimeSpan. De interfaces en de enum mogen niet worden gewijzigd.

Als je in een tweetal werk heet de ene persoon A en de andere B. Bij elke opgave staat aangegeven welke persoon die opgave moet uitwerken. De uitwerking van de opgaven van het andere groepslid moet je kritisch reviewen. Vermeld jouw naam bij @author in de aanhef van de door jou geprogrammeerde (test)klasse. Als jij de klasse hebt gespecificeerd, vermeld dan jouw naam bij de aanhef van de klasse met als toevoeging ‘SPECIFICATIE’.

**Opdrachten**

Zie volgende pagina’s.

**Week 4**

1. Schrijf unittesten voor de klassen Time en TimeSpan en haal de fouten uit de code van de klassen Time en TimeSpan.
2. Maak een unittest voor de klasse TimeSpan2 en programmeer de klasse TimeSpan2.
3. Vraag feedback aan de docent tijdens de les.

**Week 5**

1. Schrijf specificaties voor de klassen Contact en Appointment.
2. Schrijf unittesten voor de klassen Contact en Appointment.
3. Programmeer en test de klassen Contact en Appointment.
4. Laat de opdracht beoordelen door de docent tijdens de les.

**Per e-mail inleveren in week 5:**

Het gemeenschappelijke Netbeans-project van jullie groep als geheel nadat alle opdrachten zijn uitgevoerd. Zorg ervoor dat bij iedere klasse, interface en unittest de naam van de verantwoordelijke auteur (@author) vermeld staat.

**Opdrachten**

Opdracht 0 (A,B)

Bestudeer eerst het document VoorbeeldSchrijfUnittest.

Hou je bij het schrijven van een unittest aan de 2 regels:

1. Documenteer de unittesten met de zinsneden uit de specificatie van de betreffende methoden;
2. Bij het verzamelen van de testgevallen kies je ten minste 1 normale situatie en analyseer je zorgvuldig welke grensgevallen je verder moet meenemen. Dat betekent normaliter dat er meerdere testgevallen zijn per te testen operatie.

Opdracht 1 (A)

Je schrijft een unittest voor klasse Time die ITime implementeert. De klasse Time beschikt over een constructor met 5 parameters die respectievelijk naar het jaar, maand, dag, uren en minuten van het betreffende tijdstip verwijzen.

/\*\*

\* creation of a Time-object with year y, month m, day d, hours h and

\* minutes m; if the combination of y-m-d refers to a non-existing date

\* a correct value of this Time-object will be not guaranteed

\* @param y

\* @param m 1≤m≤12

\* @param d 1≤d≤31

\* @param h 0≤h≤23

\* @param m 0≤m≤59

\*/

public Time(int y, int m, int d, int h, int min)

Opdracht 2 (B)

Je schrijft een unittest voor klasse TimeSpan die ITimeSpan implementeert. De klasse TimeSpan beschikt over een constructor met twee parameters van het type ITime die de begin- en eindtijd van de TimeSpan vastlegt:

/\*\*

\* creation of a TimeSpan with begin time bt and end time et

\* @param bt begin time bt must be earlier than end time et

\* @param et

\*/

public TimeSpan(ITime bt, ITime et)

De specificatie van unionWith en intersectionWith in ITimeSpan klinkt ingewikkeld; misschien komt hij zelfs tegenstrijdig op je over. Bespreek deze specificatie met je medestudenten en vraag zo nodig toelichting van je docent.

Controleer of je je aan bovenstaande 2 regels bij het schrijven van een unittest hebt gehouden. Bijvoorbeeld bij de test van unionWith-methode in TimeSpan zou je in verband met regel 2 op zijn minst zes gevallen moeten onderscheiden. Zie onderstaande gevallen A t/m F ten opzichte van P. Eigenlijk zou je nog meer grensgevallen kunnen onderscheiden.

P

A

B

CP

E

D

F

tijdslijn

begintijd

P

eindtijd

P

Hint: onderzoek voor de zes gevallen wat, zo mogelijk, de begin- en eindtijd van de union is (kijk naar minima en maxima van de begin- en eindtijden); waaraan kun je zien dat de union niet bestaat?

Opdracht 3 (A; start nadat opdracht 1 is voltooid)

De klasse Time implementeert de interface ITime. Deze klasse bevat echter enkele programmeerfouten. Test deze klasse met de unittest uit opdracht 1 en verwijder de fouten.

Bij de declaratie van de datavelden wordt gebruik gemaakt van de klasse java.util.GregorianCalendar. Het is uitdrukkelijk NIET de bedoeling dat je een constructor met een parameter van het type GregorianCalendar opneemt. Alleen de constructor met 5 parameters uit opgave 1 is gewenst.

Hint: maak verder gebruik van de get(int)-methode in de klasse GregorianCalendar; dit is een generieke methode die van allerlei lokale constanten gebruikmaakt. Zie <http://docs.oracle.com/javase/6/docs/api/java/util/GregorianCalendar.html>. Pas op: we willen er uitdrukkelijk op wijzen dat de maandnummering binnen GregorianCalendar niet spoort met onze westerse kijk. Vergeet verder niet om zo nodig in de api van java.util.Calendar te zoeken naar geschikte instrumenten.

Opdracht 4 (B; start nadat opdracht 2 is voltooid)

De klasse TimeSpan implementeert de interface ItimeSpan. Deze klasse bevat echter enkele programmeerfouten. Test deze klasse met de unittest uit opdracht 2 en verwijder de fouten.

De klasse TimeSpan beschikt over twee datavelden van het type ITime die respectievelijk de begin- en eindtijd van het TimeSpan-object voorstellen. De klasse TimeSpan beschikt over één constructor met twee argumenten van het type ITime:

/\*\*

\* creation of a TimeSpan with begin time bt and end time et

\* @param bt bt is earlier than et

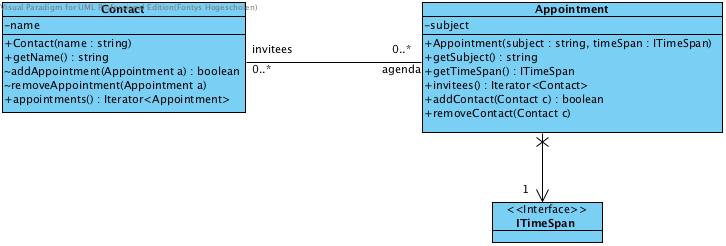
\* @param et

\*\*/

public TimeSpan(ITime bt, ITime et)

Opdracht 5 (A; start nadat opdracht 2 is voltooid)

Programmeer een alternatieve klasse TimeSpan2 die eveneens de interface ITimeSpan implementeert. De klasse TimeSpan2 beschikt over een dataveld van het type ITime die de begintijd voorstelt en een dataveld van type long die de duur van de TimeSpan voorstelt. De constructor van TimeSpan2 heeft dezelfde parameters als de constructor van TimeSpan ! Test de klasse TimeSpan2 met een kopie van de TimeSpan-unittest. Vervang in dit kopie alleen de aanroep van de constructor van TimeSpan door die van TimeSpan2.



Bestudeer bovenstaand klassendiagram met twee nieuwe klassen Contact en Appointment. Het wijzigen van de betrokken contacten bij een gegeven afspraak moet geschieden bij het Appointment-object. Omdat de agenda van het Contact-object ook moet worden bijgewerkt, is er een addAppointment- en removeAppointment met package-access aan de klasse Contact toegevoegd. Deze methoden worden alleen binnen de klasse Appointment aangeroepen. De add- en remove-method binnen de klasse Appointment hebben dus wel public access.

Opdracht 6 (B)

Specificeer (in het nederlands of engels) met behulp van javadoc een klasse Appointment met de eigenschappen:

1. 1 subject: onderwerp van het type String (mag een empty string zijn)
2. 1 TimeSpan: TimeSpane van het type ITimeSpan
3. 0 of meer genodigden van het type Contact

Er moet een manier zijn om bij een Appointment-object een Contact-object toe te voegen en te verwijderen. Het toevoegen van een genodigde mag alleen doorgang vinden indien deze afspraak niet tot een conflict in de agenda van het toe te voegen contact leidt. Neem alleen de methoden en constructoren uit bovenstaand klassendiagram in je klasse op!

Opdracht 7 (A)

Specificeer met behulp van javadoc een klasse Contact met de eigenschappen

1. name: naam van de contactpersoon
2. een agenda met Appointment-objecten waar deze contactpersoon bij is betrokken..

Aan het inspecteren van de agenda geven we vooralsnog geen aandacht. Er moet wel een manier zijn om een afspraak aan de agenda toe te voegen en te verwijderen. Conflicten (=overlap) tussen afspraken worden niet toegestaan. Omdat het toevoegen van een afspraak in eerste instantie bij de

klasse Appointment geschiedt, krijgen het toevoegen en verwijderen van afspraken hier beide package access. Neem alleen de methoden en constructoren uit bovenstaand klassendiagram in je klasse op! Let speciaal op de package-access (~) van de methoden addAppointment en removeAppointment.

Opdracht 8 (B; start nadat opdracht 6 is voltooid)

Schrijf de unittest voor de klasse Appointment.

Opdracht 9 (A; start nadat opdracht 7 is voltooid)

Schrijf de unittest voor de klasse Contact.

Opdracht 10 (B; start nadat opdracht 7 is voltooid)

Programmeer en test de klasse Contact.

Opdracht 11 (A; start nadat opdracht 6 is voltooid)

Programmeer en test de klasse Appointment.