Opgaver uge 4

Status

Hvis der er ting som i gerne vil have uddybet eller forklaret igen, så tag kontrakt til TA's eller mig

Godkendelsesopgaverne GB.1 og GB.2 skal afleveres tirsdag 7/10-2014 til instruktorerne; sammen med de andre godkendelsesopgaver GA, GC, GD og GE er deres godkendelse en forudsætning for at gå til eksamen i kurset. I må gerne arbejde sammen om godkendelsesopgaverne, men i skal aflevere individuelt og personligt kunne stå inde for besvarelserne.

Formål med opgaverne

Efter disse øvelser skal du kunne benytte teknikker til systematisk test, herunder unit test. Du skal kunne benytte JUnit, et professionelt system til automatisk kørsel af tests, som er indbygget i BlueJ.

Hvis du vil have feedback på dine besvarelser, så udskriv dem på papir og aflever dem senest ved tirsdagsøvelserne. Så retter og kommenterer hjælpelæreren dem og giver dig dem tilbage ugen efter.

Gør dette først

Aktiver understøttelse for automatisk udførelse af test i BlueJ ved at vælge TOOLS || PREFERENCES || MISCELLANEOUS || SHOW UNIT TESTING TOOLS. Dette er også beskrevet i B&K afsnit 7.4 og appendiks H.

Opvarmningsopgaver

B&K opgaver 7.13 7.14, 7.15, 7.16, 7.17, 7.18, 7.19

Opgave GB.1

Download projektet `date` fra kursets forelæsningsplan. Klassen `Date` i dette projekt gør det muligt at oprette dato-objekter som fx `new Date(2018, 10, 2)`, der repræsenterer 2. oktober 2018. Et `Date`-objekt har metoder til at finde datoens nummer i året, ugedag, ugenummer, dagnummer siden 1/1 år 1, om året er skudår, antal dage til en anden dato, og så videre.

Det er altsammen beskrevet i `Date`-klassens interface. For at løse denne opgave *behøver du kun se på klassens interface* (Documentation), *ikke dens implementation* (Source Code).

Denne opgave går ud på at lave unit test til klassen `Date`.

(i) Opret en testklasse `TestDate`, højreklik på den og vælg CREATE TEST METHOD for at bruge test-optageren. Du skal lave et testtilfælde, fx kaldet `testTue20181002`, der tjekker om 2. oktober 2018 beregnes til at være en tirsdag, dvs. ugedagsnummer 1. Dette kræver at du opretter et `Date`-objekt for 2013-10-02, kalder dets `weekDay()`-metode, siger at resultatet skal være 1, og klikker RECORDING || END. Kig på klasse `TestDate` i editoren og se hvordan kroppen af test-metoden `testTue20181002` ser ud. Oversæt `TestDate` og kør testen.

(ii) Optag på samme måde en test, fx kaldet `test20180101`, som opretter et objekt svarende til 1. januar 2018 og tjekker at ugedagen er mandag, altså 0; at ugenummeret er 1; at dagens nummer i året er 1; og at året ikke er skudår. Kig på den genererede testmetode `test20180101` i editoren når optagelsen er færdig.

(iii) Optag på samme måde endnu en test der tjekker de samme metoder på en anden dato, fx 31. december 2018 (som er mandag, uge 53, dag nummer 365 i året).

(iv) Brug editoren til at skrive en testmetode, fx kaldet `testDaynumbers2018`, der undersøger nogle dagnumre i år 2018. For eksempel kan du tjekke at 31. januar er dag nummer 31, at 1. februar er dag nummer 32, at 28. februar er dag nummer 59, at 1. marts er dag nummer 60, at 1. december er dag nummer 335 og at 31. december er dag nummer

365.

(v) Skriv en testmetode, fx kaldet `testYearEnd`, der for alle årene 1582 til 2500 tjekker at der er netop 1 dag fra 31. december til 1. januar det følgende år. Vink: En `for`-løkke vil vise sig nyttig. For hvert testår skal du lave to `Date`-objekter og finde antal dage mellem dem.

(vi) Skriv en testmetode, fx kaldet `testYearLength`, der for alle årene 1582 til 2500 tjekker at der er 365 dage fra 1. januar til 31. december hvis året er et skudår, og 364 dage hvis året ikke er et skudår.

(vii) Skriv en testmetode, fx kaldet `testFebMar`, der for alle årene 1582 til 2500 tjekker at der er 2 dage fra 28. februar til 1. marts hvis året er et skudår, og 1 dag hvis året ikke er et skudår.

(viii) Hvor meget tillid har du nu til at `Date`-klassen virker? Kan den pålideligt bruges til at beregne hvilken ugedag du er født på? Ugedagene dine bedsteforældre er født på? Tror du på programmet når det siger at 1. januar 2100 er en fredag (ugedag 4)? Hvad mere skal testes for at du vil vædde 1000 kroner på at klassen virker korrekt?

Aflever en udskrift af test-klassen.

Opgave GB.2

Hvis du nu studerer implementationen af klasse `Date`, så viser det sig at metoden `dayInYear()` kaldes ofte. For eksempel kaldes den direkte eller indirekte af `dayNumber`, `weekDay`, `weekNumber` og `daysTill`.

Ulrik Funder, der er en gammel hacker, kommer forbi og hævder at det er dumt at `dayInYear()` indeholder en `for`-løkke og derfor kan være noget langsom, når den kaldes så ofte. Han hævder også at nedenstående udgave af `dayInYear()`, som ikke indeholder en løkke, giver lige så gode resultater:

```
public int dayInYearNew() {
  int m = (month+9)%12, janfeb = (isLeapyear() && month >= 3 ? 60 : 59);
  return (m/5 * 153 + m%5 * 30 + (m%5 + 1)/2 + janfeb) % 365 + day;
}
```

Men denne metode ser svært kryptisk ud og det er ikke spor klart hvordan den fungerer (og Ulrik er lidt afdanket), så vi er skeptisk indstillet.

Prøv at indsætte denne nye version af `dayInYear` i klasse `Date` (og fjern eller omdøb den gamle metode) og kør alle dine tests igen. Havde Ulrik ret i at den nye metode virker lige så godt som den gamle?

Aflever en 5-10 linjers besvarelse i ord.

Opgave 6.3 — ekstraopgave: hvis du mangler tid, så drop den

Denne delopgave viser at man sagtens, og med fordel, kan skrive tests *inden* man skriver implementationen af en metode, hvis bare man har en god beskrivelse (dokumentation) af metoden.

Klasse `Date` har en metode `Date next()`, hvis dokumentation siger at den skal returnere et `Date`-objekt der repræsenterer den næste dag. Nogle eksempler:

Hvis `date = new Date(2018, 10, 2)` så skal `date.next()` repræsentere 3. oktober 2018.
Hvis `date = new Date(2018, 10, 31)` så skal `date.next()` repræsentere 1. november 2018.
Hvis `date = new Date(2018, 12, 31)` så skal `date.next()` repræsentere 1. januar 2008.

Men metodens *implementation* er tydeligvis ikke korrekt: den returnerer altid juleaften 1945.

(i) Skriv nogle test-metoder i klasse `TestDate` som tester om metoden `next()` svarer til dokumentationen. Vink: For eksempel burde der naturligvis være 1 dag fra en given dag til den næste dag, så der skal gælde at `date.daysTill(date.next())` er lig med 1, uanset hvad `date` er. Din testmetode kan enten tjekke dette for nogle få velvalgte datoer, eller du kan skrive en `for`-løkke til at gøre det for nogle tusind dage.

Kør testen. Det er klart at den vil fejle på grund af den forkerte implementation af `next()`.

(ii) Skriv en bedre implementation af metode `next()`. Læg mærke til at metoden skal lave et nyt `Date`-objekt og returnere det, ikke ændre det givne objekt. Vink: Brug nogle `if-else` til at skelne tre tilfælde: (a) ikke sidste dag i måneden; (b) sidste dag i måneden men ikke sidste dag i året; og (c) sidste dag i året.

I tilfælde (a) skal `day` i det nye `Date`-objekt være 1 større end i det gamle objekt; i tilfælde (b) skal måneden være 1 større og dagen sættes til 1; og i tilfælde (c) skal året være 1 større, og dag og måned skal sættes til 1. Brug tabellen `monthLength` til at afgøre om en dag er den sidste i måneden. Til at begynde med kan du ignorere skudår — i skudår skal er dagen efter 28. februar jo ikke sidste dag i måneden.

Kør din test på din nye `next)()` metode og se om den virker korrekt.

Aflever en udskrift af testen af `next()`, og en udskrift af `next()`-metoden implementation, samt en diskussion (som programkommentar) om hvorvidt implementationen klarer alle testene.