COMPUTERSPIL 2

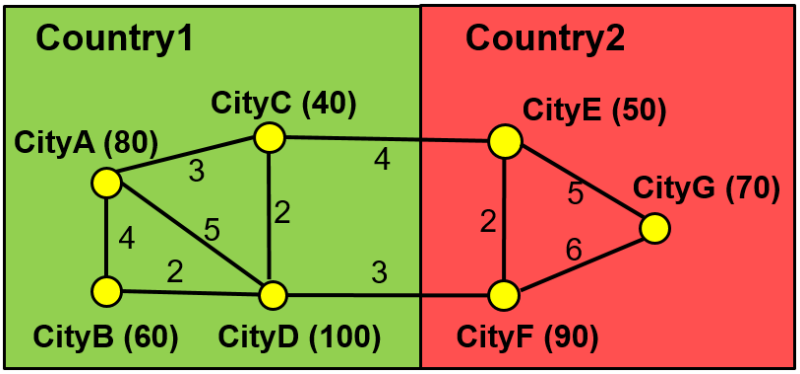
I denne anden delaflevering skal I bruge de ting, som I har lært om automatiske regression tests, på de klasser, som I har implementeret i den første delaflevering. Herudover skal I rette de fejl og mangler, som instruktoren har påpeget i jeres første delaflevering.

I BlueJ bogens kapitel 9 og Appendix G er det beskrevet, hvordan man bør afteste sine klasser, herunder hvordan man (ved hjælp af JUnit testing frameworket) laver automatiske regression tests, der let kan gentages, hver gang man har modificeret sin kode. Bemærk at det er langt hurtigere (og nemmere) selv at skrive Java koden for de enkelte tests (i stedet for at optage testene interaktivt).

I behøver ikke at lave testmetoder for trivielle accessor og mutator metoder, der blot returnerer/ændrer en enkelt feltvariabel uden at gøre andet, og I behøver heller ikke at lave testmetoder for ***compareTo***, ***equals*** og ***hashCode*** metoderne. Det bør man normalt gøre, men da det kan være lidt besværligt og tidskrævende, har vi besluttet at lade jer slippe for det.

Dokumentationen for jeres testmetoder kan holdes på et minimum. Testmetoder har ingen parametre og returnerer intet, så @param og @return tags giver ikke mening. Herudover vil testmetodens navn normalt fortælle, hvilken metode den tester. I kan derfor nøjes med at lave @author og @version tags for testklasserne samt indsætte // kommentarer udvalgte steder i jeres kode.

I jeres aftestning kan I bruge den Test Fixture, der er indeholdt i klassen ***CGTest***. Den indeholder et ***Game*** objekt, to ***Country*** objekter, syv ***City*** objekter og et antal ***Road*** objekter, der realiserer nedenstående netværk, hvor tallene i parenteserne angiverne byernes initiale værdi, mens tallene på vejene angiver deres længde:



Test Fixturen kan kopieres til jeres egne testklasser via følgende skridt:

- Højreklik på ***CGTest*** og vælg *"Test Fixture to Object Bench"*.
- Højreklik på den ønskede testklasse og vælg *"Object Bench to Test Fixture"*.

Ved afslutningen af hver opgave, kan I afprøve, om det, som I har lavet, fungerer korrekt. Dette gøres ved at kalde klassemetoden ***test*** i ***TestServer*** klassen med parameteren "CG2-X", hvor X er nummeret på den pågældende opgave. Det afprøves, at jeres regression tests:

- *ikke* finder fejl i et korrekt projekt,
- finder *alle* "oplagte" fejl i nogle forkerte projekter.

Opgave 1

I en forelæsningen blev der vist et eksempel på en testklasse ***RoadTest***, der tester konstruktøren og de ikke trivielle metoder i ***Road*** klassen. Kopiér indholdet af testklassen til jeres projekt, og udfyld de manglende dele.

Jeres regression tests skal kontrollere, at konstruktøren initialiserer alle feltvariabler korrekt. Herudover skal de kontrollere at ***toString*** metoden returnerer en tekststreng på den specificerede form. Som nævnt tidligere, behøver I ikke at lave regression tests for ***compareTo***, ***equals*** og ***hashCode*** metoderne.

Udfør testene ved at højreklikke på ***RoadTest*** klassen og vælge *TestAll*, eller ved at trykke på knappen *Run Tests* i venstre side af vinduet med BlueJ's klassediagram (hvis knappen ikke er synlig trykkes først på den lille trekantede knap).

Hvis man bruger *Run Tests* knappen, skal man huske at tjekke, om nogle af testklasserne har skrå grå striber i den nederste del, hvilket viser, at de skal kompileres før man kan udføre dem (ellers ignoreres de pågældende testklasser).

Hvis en eller flere tests fejler, skal I forsøge at finde ud af, hvad der er galt, og rette dette. Det kan både være jeres implementation af ***Road*** klassen og jeres implementation af testene i ***RoadTest*** klassen, der er forkert. Men hvis testdriveren tidligere har godkendt jeres ***Road*** klasse er det mest sandsynligt, at fejlen ligger i ***RoadTest***.

Opgave 2

I en forelæsning blev der vist et eksempel på en testklasse ***PositionTest***, der tester konstruktøren og nogle af de ikke trivielle metoder i ***Position*** klassen. Kopiér indholdet af testklassen til jeres projekt, og udfyld de manglende dele.

Jeres regression tests skal kontrollere, at konstruktøren initialiserer alle feltvariabler korrekt. Herudover skal de kontrollere at ***move***, ***turnAround***, ***hasArrived*** og ***toString*** metoderne fungerer korrekt, dvs. foretager de forventede opdateringer af feltvariablerne og har korrekte returværdier. Husk at afprøve metoderne i de forskellige situationer, der kan opstå, inklusiv diverse grænsetilfælde, f.eks. at feltvariablen ***distance*** er lig 0.

Som nævnt tidligere, behøver I ikke at lave regression tests for ***equals*** og ***hashCode*** metoderne.

Udfør testene ved at højreklikke på ***PositionTest*** klassen og vælge *TestAll*, eller ved at trykke på knappen *Run Tests* i venstre side af vinduet med BlueJ's klassediagram. Hvis en eller flere tests fejler, skal I forsøge at finde ud af, hvad der er galt, og rette dette. Hvis testdriveren tidligere har godkendt jeres ***Position*** klasse, er det mest sandsynligt, at fejlen ligger i ***PositionTest***.

Opgave 3

Lav en testklasse ***CityTest*** med velvalgte regression tests for ***City*** klassen. Det er vigtigt, at I både tester, hvad der normalt sker, og hvad der sker i specialtilfælde og tæt ved grænseværdier. Husk også, at der kan være både positive og negative tests.

Når I skal teste ***arrive*** metoden i ***City*** klassen kan det, som vist i en forelæsning, være nyttigt at kunne kontrollere den seed værdi som ***Random*** objektet bruger. På den måde kan I finde ud af, hvad ***bonus*** metoden returnerer, når den kaldes inde fra ***arrive*** metoden.

Jeres regression tests skal kontrollere, at konstruktøren initialiserer alle feltvariabler korrekt. Herudover skal de kontrollere at ***changeValue***, ***reset***, ***toString*** og ***arrive*** metoderne fungerer korrekt, dvs. foretager de forventede opdateringer af feltvariablerne og har korrekte returværdier. Husk at afprøve metoderne i de forskellige situationer, der kan opstå, inklusiv diverse grænsetilfælde, f.eks. hvad der sker med ***arrive*** metoden, når feltvariablen ***value*** er 0.

Som nævnt tidligere, behøver I ikke at lave regression tests for ***compareTo***, ***equals*** og ***hashCode*** metoderne.

Udfør testene ved at højreklikke på ***CityTest*** klassen og vælge *TestAll*, eller ved at trykke på knappen *Run Tests* i venstre side af vinduet med BlueJ's klassediagram. Hvis en eller flere tests fejler, skal I forsøge at finde ud af, hvad der er galt, og rette dette. Hvis testdriveren har godkendt jeres ***City*** klasse, er det mest sandsynligt, at fejlen ligger i ***CityTest***.

Opgave 4

Lav en testklasse ***CountryTest*** med velvalgte regression tests for ***Country*** klassen. Det er vigtigt, at I både tester, hvad der normalt sker, og hvad der sker i specialtilfælde og tæt ved grænseværdier. Husk også, at der kan være både positive og negative tests.

Når I tester ***bonus*** metoden, skal I kalde metoden mange gange og derefter tjekke at middelværdien er som forventet, samt at alle de forventede værdier returneres (og ingen andre). Det svarer helt til det, som I gjorde, da I testede ***roll*** metoden i Raflebæger 4. Som vist i en forelæsning, bør testet gentages for mange forskellige seed værdier. I bør også teste grænsetilfældene, hvor ***bonus*** kaldes med parameterværdierne 0 og 1.

Jeres regression tests skal kontrollere, at konstruktøren initialiserer alle feltvariabler korrekt. Herudover skal de kontrollere at ***getCities***, ***getCity***, ***getRoads***, ***reset***, ***toString***, ***bonus, addCity, addRoads, position og readyToTravel*** metoderne fungerer korrekt, dvs. foretager de forventede opdateringer af feltvariablerne og har korrekte returværdier. Husk at afprøve metoderne i de forskellige situationer, der kan opstå, inklusiv diverse grænsetilfælde, herunder specielt dem, der er nævnt i den forklarende tekst under de forskellige metoder.

I testene af konstruktøren og metoder som ***getCities*** og ***addRoads***, er det tilstrækkeligt, at I tjekker antallet af byer/veje. I behøver ikke at tjekke, at det er de rigtige byer/veje, der ligger i network/returneres.

Som nævnt tidligere, behøver I ikke at lave regression tests for ***equals*** og ***hashCode*** metoderne.

Udfør testene ved at højreklikke på ***CountryTest*** klassen og vælge *TestAll*, eller ved at trykke på knappen *Run Tests* i venstre side af vinduet med BlueJ's klassediagram. Hvis en eller flere tests fejler, skal I forsøge at finde ud af, hvad der er galt, og rette dette. Hvis testdriveren tidligere har godkendt jeres ***Country*** klasse, er det mest sandsynligt, at fejlen ligger i ***CountryTest***.

Opgave 5

I skal nu afprøve om de ting, som I har lavet Computerspil 2, fungerer korrekt ved at kalde klassemetoden ***test*** i ***TestServer*** klassen med parameteren "CG2". Det afprøves, at jeres regression tests:

- *ikke* finder fejl i et korrekt projekt,
- finder de *fleste* fejl i nogle forkerte projekter.

Hvis testserveren finder fejl, skal l gennemgå jeres kode og forsøge at rette dem.

I de kommende delafleveringer, vil I modificere jeres kode. Når I gør det, skal I opdatere jeres dokumentation og jeres regression tests. Når I tilføjer nye klasser og nye metoder/konstruktører, skal I forsyne disse med dokumentation og regression tests.