COMPUTERSPIL 3

I denne tredje delaflevering skal I bruge nogle af de ting, som I har lært om nedarvning og "dynamic method lookup" til at strukturere jeres kode, når vi indfører flere forskellige slags lande og flere forskellige slags byer. Herudover skal I rette de fejl og mangler, som instruktoren har påpeget i jeres anden delaflevering.

Husk at holde jeres dokumentation og regression tests opdaterede, så de afspejler de ændringer/ tilføjelser, som I laver i jeres kode. Når I laver nye klasser og nye metoder/konstruktører, skal I lave dokumentation og regression tests for disse. I behøver dog ikke at teste konstruktørerne i de nye klasser. Der tilføjes ikke nye feltvariabler, hvorfor konstruktørerne er ret trivielle.

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 "CG3-X", hvor X er nummeret på den pågældende opgave. Afprøvningen består af to ting:

- Nogle regression tests for konstruktørerne og metoderne i jeres klasser (som i Computerspil 1).
- Afprøvning af jeres egne regression tests (som i Computerspil 2).

Opgave 1

Implementér ***BorderCity*** klassen, der repræsenterer en grænseby. Klassen skal være en subklasse af ***City*** klassen. Husk at konstruktøren skal kalde superklassens konstruktør via det reserverede ord ***super***.

Subklassen skal overskrive superklassens ***arrive*** metode (jvf. opgave 5 i Computerspil 1), således at spilleren, udover at modtage den normale bonus, også skal betale told, såfremt spilleren ankommer fra et andet land (hvilket afgøres ved hjælp af ***equals*** metoden). Tolden udgør en procentdel af spillerens formue inden tillæg af bonus (nedrundet til nærmeste heltal).

For at kunne finde spillerens formue og det land, som spilleren kommer fra, er det nødvendigt, at ***BorderCity*** klassens ***arrive*** metoden har et ***Player*** objekt som parameter. Da metoden kun kan overskrive en metode, der har de samme parametre, skal I tilføje nedenstående metode til ***City*** klassen:

- ***public int arrive (Player p) { return arrive(); }***

Som det ses kalder den nye ***arrive*** metode blot den gamle (og ignorerer sin parameter).

I ***BorderCity*** klassens ***arrive*** metode anvendes den nye parameter til at kalde accessor metoderne ***getMoney*** og ***getFromCountry*** i ***Player*** klassen. Den første metode returnerer spillerens formue, mens den anden returner det land, som spilleren kommer fra.

Den procentdel af formuen, der skal betales i told, kan hentes via nedenstående metodekald, der returnerer et heltal i intervallet [0,100].

- ***getCountry().getGame().getSettings().getTollToBePaid()***

Den betalte told adderes til byens værdi, som dermed forøges. Bonus minus told returneres som returværdien af ***arrive*** metoden.

Til aftestning af ovenstående metoder, kan I bruge den Test Fixture, der stilles til rådighed via ***CGTest*** klassen (beskrevet i Computerspil 2). I skal blot ændre klassens ***setUp*** metode, således, at ***CityC***, ***CityD***, ***CityE*** og ***CityF*** kommer til at tilhøre klassen ***BorderCity*** i stedet for klassen ***City***.

Som vist i en forelæsning, skal l lave to testmetoder for ***arrive*** metoden i ***BorderCity***. Den første tjekker, at der betales korrekt told (dvs. 20%), når spilleren kommer fra et andet land, mens den anden tjekker, at der ikke betales told, når spilleren kommer fra samme land.

Opgave 2

Implementér ***CapitalCity*** klassen, der repræsenterer en hovedstad. Klassen skal være en subklasse af ***BorderCity*** klassen (idet alle hovedstæder i Norden har direkte forbindelser til udlandet). Husk at konstruktøren skal kalde superklassens konstruktør via det reserverede ord ***super***.

Subklassen skal overskrive superklassens ***arrive*** metode, således at spilleren udover at modtage bonus (og eventuelt betale told), bruger en del af sin formue (idet hovedstæder har mange dyre fristelser). Forbruget beregnes som et tilfældigt heltal i intervallet ***[0,money]***, hvor ***money*** er spillerens formue (efter tillæg af bonus og fradrag af eventuel told). Husk at anvende det ***Random*** objekt, der er skabt af ***Game*** objektet. Dette kan hentes via ***getGame*** metoden i ***Country*** klassen.

De forbrugte penge adderes til byens værdi, som dermed forøges. Bonus minus told og forbrug returneres som returværdien for ***arrive*** metoden.

Til aftestning af ovenstående metoder, kan I bruge Test Fixturen fra ***CGTest***. I skal blot ændre klassens ***setUp*** metode, således, at ***CityD*** og ***CityE*** bliver hovedstæder.

I skal lave to testmetoder for ***arrive*** metoden i ***CapitalCity***. Den første tjekker, hvad der sker når spilleren kommer fra et andet land, mens den anden tjekker, hvad der sker, når spilleren kommer fra samme land. Testene ligner de to testmetoder fra ***BorderCity***, idet I dog nu også skal beregne, hvor mange penge spilleren bruger, og tage hensyn til dette i jeres assertions.

Opgave 3

Implementér ***MafiaCountry*** klassen, der repræsenterer et land, hvor man har risiko for at blive berøvet hver gang man besøger en by. Klassen skal være en subklasse af ***Country*** klassen. Husk at konstruktøren skal kalde superklassens konstruktør via det reserverede ord ***super***.

Subklassen skal overskrive superklassens ***bonus*** metode, således at man risikerer at blive berøvet i stedet for at få en bonus. Dette opnås ved at subklassens ***bonus*** metoden nogle gange kalder superklassens ***bonus*** metode, mens den ellers returnerer et negativt tal, der angiver, hvor meget spilleren mister ved røveriet.

Den procentvise risiko for at blive røvet (angivet som et heltal) kan hentes via metodekaldet:

- ***getGame().getSettings().getRisk()***

mens tabet ved røveriet kan hentes (som et positivt heltal) via metodekaldet:

- ***getGame().getLoss()***

En spiller kan selvfølgelig ikke miste flere penge end hun pt. har. Det sørger ***Game*** klassen for, så det behøver I ikke at bekymre jer om.

Sverige er et Mafialand, hvilket ses ved, at det er farvet rødt. Undskyld til svenskerne for dette påhit!

Til aftestning af ovenstående metoder, kan I bruge Test Fixturen fra ***CGTest***. I skal blot ændre klassens ***setUp*** metode, således at ***country2*** bliver et Mafialand.

Testmetoden for ***bonus*** metoden i ***MafiaCountry*** kan udformes som den tilsvarende testmetode i ***Country*** klassen. I skal tjekke, at spilleren bliver røvet ca. 20 % af gangene, at tabet i gennemsnit udgør 30€, og at det kan antage alle heltalsværdier i intervallet [10,50]. Hvis I ønsker at ændre risikoen for røveri, kan dette gøre via *Options* knappen. Herudover skal I teste, at bonussen beregnes korrekt, når man ikke bliver røvet.

Opgave 4

Som beskrevet i forrige opgave, kan bonussen, ved ankomsten til en by i et Mafialand, blive negativ (hvilket indikerer, at spilleren har været udsat for røveri).

Modificér ***arrive*** metoden i ***City*** klassen. Hvis ***bonus*** metoden returnerer en positiv værdi, skal alt fungere som før, men hvis værdien er negativ, returneres denne uden at ændre byens værdi (idet de røvede penge går i Mafiaens lommer og dermed er tabt for spillet).

Undersøg om der også er behov for at ændre ***arrive*** metoderne i ***BorderCity*** og ***CapitalCity*** og implementér eventuelle ændringer, der er nødvendige.

Modificér testmetoden for ***arrive*** i ***CityTest***, således at den tager hensyn til ovenstående og tester metoden på to byer, hvoraf den ene ligger i et Mafialand, mens den anden ikke gør det.

Opgave 5

I skal nu afprøve om de ting, som I har lavet Computerspil 3, fungerer korrekt, men først skal nogle af projektets klasser opdateres, hvilket sker ved at I kalder klassemetoden ***download*** i ***TestServer*** klassen med parameteren ”CG3”.

Kald klassemetoden ***test*** i ***TestServer*** klassen med parameteren ”CG3”. Det er kun de nye klasser, der testes, og afprøvningen består af to ting:

- Nogle regression tests for konstruktørerne og metoderne i jeres klasser (som i Computerspil 1).
- Afprøvning af jeres egne regression tests (som i Computerspil 2).

Testserveren forudsætter, at I ikke laver unødvendige kald af metoderne i ***Random*** objektet. Hvis f.eks. ***bonus*** metoden i ***MafiaCountry*** kalder ***getLoss*** for a beregne tabet i en situation, hvor spilleren slet ikke bliver røvet, vil de pseudo-tilfældige tal komme ud af sync, og testen vil fejle.

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

Afprøv spillet

Spil nogle spil og afprøv de forskellige faciliteter i spillet. Prøv at finde nogle gode strategier for at samle flest mulige penge sammen. Hvor dyrt er det at rejse fra et land til et andet, og hvor vigtigt er det at undgå hovedstæder og mafialande?