Handin 3

Simon Stenbæk Madsen - 20102187
Thomas Philkjær - 20092289

Test List:

EpsilonCiv

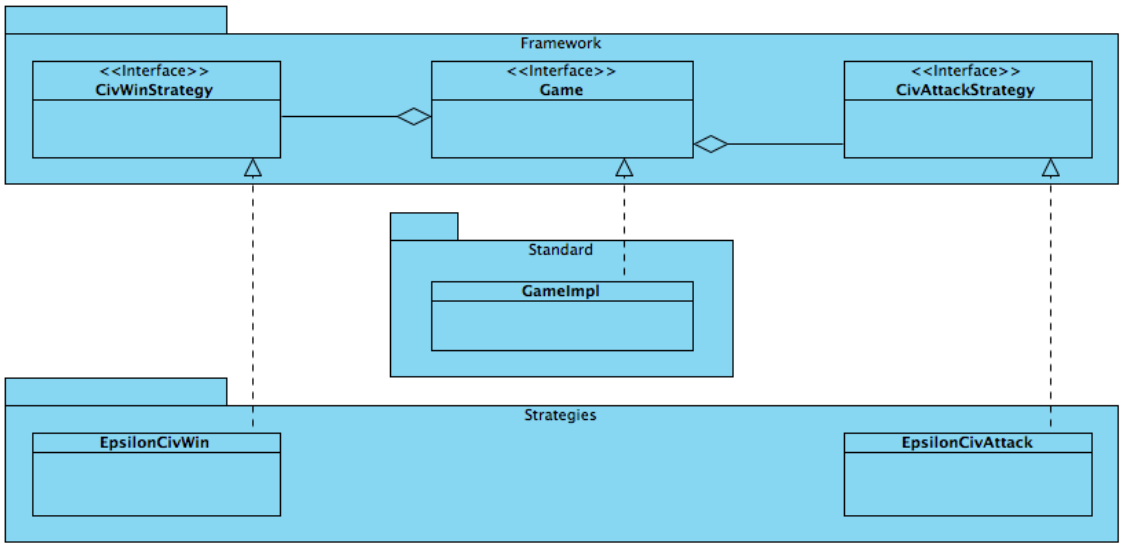
Attacker gets deleted from board/world after attacker loses
Test: attackerGetsDeletedFromWorldAfterLoss()
After 3 succesfull attacks Red wins
Test: redShouldWinAfter3SuccessfulAttacks()
After 3 succesfull attacks Blue wins
Test: blueShouldWinAfter3SuccessfulAttacks()
Attack strength of Legion placed on a Hill who attacks an archer
Test: attackStrengthOfLegionAttackingArcherIn0x1()
Attack strength of Legion who attacks an archer in a forest
Test: attackStrenthOfLegionAttackingArcherInForestAt9x1()
Attack strength of Legion placed in a City who gets attacked by an archer
Test: blueAttackingArcherShouldDieToLegionInCityAt8x12()
Red Settler cannot kill blue legion
Test: redSettlerAt4x3ShouldNotKillBlueLegionAt3x2WhenSettlerIsAttacking()
Red archer uses red archer on adjacent tiles to defeat blue legion
Test: attackStrengthOfRedArcherWithFriendOnAdjacentTile()
The dice factor should multiply a unit strength by a factor of 2
Test: diceFactorShouldMultiplyStrengthByTwo()

ZetaCiv

When 20 turn has past you after 3 succesfull attacks
Test:after20TurnsYouWinByTaking3SuccessiveAttacks()
Win by taking all the cities before 20 rounds has past
Test: after17TurnsYouWinByTakingAllTheCities()
After 20 rounds you do not win by taking all the cities
Test:after20TurnsYouDoNotWinByTakingAllTheCities()
No one automatic wins after 20 rounds()
Test: noOneAutomaticWinsAfter20Rounds()

EpsilonCiv:

Grundet genbrug af CivWinStrategy og CivAttackStrategy ser at vores uml diagram ser sådan som det nu gør.



ZetaCiv ændrer sejrs-grundlaget sig alt afhængigt af tiden derfor har vi lavet en klasse der implementerer CivWinStrategy. Tiden bliver styret ved hjælp af endofturn-metode nok gange. Dette gøres for at teste funktionen.
BetaCivWin bruges hvis der går under 20 turns. Hvis der går mere end 20 turns så er det EpsilonCIvWin der bruges ved hjælp af et tjek der holder styr på om de første20 turns der gået.

36.17 Refaktorisering
Refaktoriserings processen har for det meste bestået af gameImpl konstuktøren pga factory pattern. Udover det er der ikke refaktoriseret noget. At lave denne refaktorisering giver lav kopling imellem GameImpl og de forskellige varianter. Ved at bruge AbstractFactory forbliver antallet af konstruktøren konstant også selvom vi ændrer på varianter hvilket formindsker risikoen for at noget gå i stykker. Ligeledes gør brugen af AbstractFactory det nemmere at oprette nye varianter da der er en specifik kontrakt med AbstractFactory-interfac'et.

Angående teststub'en(FixedTestStubCivDieRoll) har vi kun brugt det for at komme uden om afhængigheden af metoden random der som bekendt giver et tilfældigt resultat. Vi lavede en simple implementation som returnerede en konstant for at kunne verificerer den kode som bruger terningkastet. Dette har resulteret i test-specifikke udgaver af factory-varianter hvor terningkast bruges.

36.18 Test stub

AlphaCivWorld(DependentOnUnit) er en statisk implementation af CivWorldStrategy interfacet. Dette betyder at man kan bruge direct input til at verificere GameImpl(UnitUnderTest). AlphaCiv (CivWorldStrategy) er jf. definitionen en dependent on unit el. test stub fordi implementationen er så simpel at den direkte kan verificeres, og pga. dens implementation af et fælles interface kan substituere enhver implementation af CivWorldStrategy.