**Tenebris**

**Grupp 24**

**Kodgranskning**

**V. 1.3**

**2018-04-06**

# **Dokumenthistorik**

|  |  |  |  |
| --- | --- | --- | --- |
| Datum | Version | Beskrivning | Författare |
| 180406 | <1.0> | Skapande av dokumentet. | Anton Persson |
| 180407 | <1.1> | Processbeskrivning, checklista och riktlinjer | Anton Persson |
| 180411 | <1.2> | Granskningsprotokoll | Anton Persson, Dionist Peci |
| 180525 | <1.3> | Andra granskningsprotokellet | Anton Persson, Dionist Peci |
|  |  |  |  |
|  |  |  |  |

**Innehåll**

[**Dokumenthistorik**](#_gjdgxs) **2**

[**Kodgranskning**](#_30j0zll) **4**

[**Syfte**](#_1fob9te) **4**

[**Ordlista**](#_3znysh7) **4**

[**Referenser**](#_2et92p0) **4**

[**Process**](#_tyjcwt) **5**

[**Riktlinjer för kod**](#_3dy6vkm) **6**

[**Granskningsprotokoll**](#_2s8eyo1) **6**

# **Kodgranskning**

# **Syfte**

Syftet är att dokumentera granskningen av koden - inkluderat processbeskrivning, granskningsresultat, och ordlista.

# **Ordlista**

<if-sats> <En stycke av kod som körs om ett eller flera villkor uppfylls.>

<Camel-case> <är ett sätt att skriva samman ord utan bindestreck eller mellanslag, men med inledande versal på varje ingående orddel.>

<loop> <En del kod i programmering som körs om och om igen enligt ett visst/vissa villkor.>

<Klass> <En del kod inom programmering som representerar någon slags entitet, kontroll för logik, eller användargränssnitt.>

<Konstruktor> <En speciell metod som körs för att skapa ett objekt.>

<Metod> <En del kod som utför ett antal instruktioner. Har ett tilldelat namn och parameterlista.>

<Variabel> <En variabel har någon slags värdetyp och ett visst värde som kan ändras genom programmets gång. Variabler har även namn.>

# **Referenser**

[1] Tsui, F., Karam, O., Bernal, B. (2014) Software Engineering. Jones & Bartlett Learning. S220-221.

[2]Dr.Benjamin Hummel, *Change-based vs. File-based Code Reviews,* https://www.cqse.eu/en/blog/change-based-vs-file-based-reviews/, hämtad (2018-04-06).

# **Process**

Metoden som använts för kodgranskningen av Tenebris är file-based review som förklarat enligt Dr.Benjamin Hummel under konstruktionen av ConQAT - år 2005[2]. Granskningen kommer även hanteras enligt de sex stegen som beskrivs i Essentials of Software Engineering[1] av Tsui men nedskalat till fem punkter som en anpassning av projektstorleken. Dessa metoder är anpassade efter minimala verktygs-krav samt lågt krav av resurser, vilket passar projektets generella struktur.

Granskningen innehåller tre roller: Moderator (Leder mötet, bestämmer relevant kod samt tar hand om avslutning och start av granskningen), Sekreterare (Skriver protokoll samt uppdaterar markeringar), samt Deltagare (alla deltagande i granskningsprocessen. Har hand om den individuelle granskning av koden enligt checklistan).

Alla relevanta (väljs av Moderatorn, alltså den ansvarige för kodgranskningen) filer i Tenebris kommer markeras antingen som röda: (i utveckling eller misslyckad granskning), gula (redo för granskning), eller grön (granskning slutförd) under planeringsfasen inför varje kodgranskning, och sedan sorterad in i prioriteringsgrader (dessa används som stödmaterial till kodgranskningen). Mötet kommer sedan att struktureras i följande ordning:

**1. Planera**

Moderatorn väljer relevant kod inför granskningsmötet baserat på ovannämnda markeringar samt prioriteringen. Moderatorn inleder sedan mötet och informerar alla deltagande.

**2. Förberedelse**

Varje deltagare granskar enskilt det som ska granskas med hänsyn till checklistan.

**3. Inspektion**

En gemensam kort granskning genomförs där prioritering och markering uppdateras, och sedan delas defekterna ut mellan alla närvarande.

**4. Omarbetning**

Defekterna som upptäcks åtgärdas enligt tilldelningen.

**5. Uppföljning**

Moderatorn kontrollerar omarbetningen.

# **Riktlinjer för kod**

1. Alla kommentarer skrivs på Engelska.
2. Alla klasser börjar med stor bokstav och följer sedan camel-case.
3. Alla variabler börjar med liten bokstav och följer sedan camel-case.
4. Alla metoder börjar med stor bokstav och följder sedan camel-case om namnet bildar en mening.
5. All namngivning ska vara tillräckligt beskrivande (endast förkortningar om det är en självklar betydelse).
6. Det ska alltid finnas en rad mellan varje kodblock (metoder m.m).
7. Alla klasser struktureras i följande ordning variabler, konstruktor, metoder.
8. Alla metoder, if-satser, och while-looper ska skrivas på följande sätt:

if-sats/metod/loop

{

Kod

}

**Checklista för granskningsmöte**

1. Följer koden riktlinjerna?
2. Finns där redundans i koden?
3. Innehåller klassen endast relevant kod?
4. Är koden välstrukturerad och konsekvent?
5. Överensstämmer koden med kommentarerna?
6. Är varje metod förståelig och korrekt kommenterad?
7. Finns det oanvända kod stycken?

# **Granskningsprotokoll**

**Granskningsprotokoll KG1**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Fil/Klass | Rad | Defekt | Prioritering | Ansvar | Korrigerat |
| Alla | \* | Finns inga kommentarer | Medel | Dionist | Y |
| Game1.cs | 82-106 | Fel placering av kodstycke | Hög | Dionist | Ý |
| Game1.cs | 14-19 | Fel namngivning av variabel | Medel | Anton | Y |
| Green.cs | 48 - 70 | Upprepning av if-sats | Hög | Anton | Y |

**Granskningsprotokoll KG2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Alla | \* | Omvandla alla public variabler till properties (get/set) | Hög | Anton | Y |
| Alla | \* | Reducera redundans | Hög | Anton | Y |
| Alla | \* | Korrekt Accessorer | Hög | Anton | Y |
| Mage.cs | 135 - 160 | “Dubblering” av kod | Medel | Dionist | Y |
| Alla | \* | Ändra alla otydliga variabel namn | Medel | Dionist | Y |