T-302-HONN: Project 1
Design Patterns and Design Principles

Þórður Friðriksson
thordurf@ru.is

Háskólinn í Reykjavík — Reykjavík University

Inngangur

Takið eftir: Vinsamlegast lesið yfir alla eftirfarandi punkta áður en þið byrjið á verkefninu

1. **Hópar**
   - Þetta verkefni getur verið skilað í 1-4 manna hópum
   - Þið þurfið eingöngu að skila verkefninu einu sinni per hóp
   - Munið að að merkja verkefnið á forsíðu með nafni og netfangi á öllum hópmeðlimum
2. **Reglur um svindl**
   - Lausnin verður að vera ykkar eigið verk, ef upp kemst um svindl fá allir tengdir aðilar 0 fyrir verkefnið ef þetta er fyrsta brot, ef þetta er endurtekið brot má búast við harðari refsingu. Sjá reglur skólans um verkefnavinnu: ru.is/namid/reglur/reglur-um-verkefnavinnu
3. **Verkefnaskil**
   - Verkefnum skal vera skilað á Canvas bæði sem pdf skrá fyrir öll skrifleg svör og sem zip skrá fyrir kóðann
   - pdf skráin skal heita: `{student1@ru.is}-{student2@ru.is}-project1.pdf`
   - zip skráin skal heita: `{student1@ru.is}-{student2@ru.is}-project1.zip`
   - **Ef ekki er fylgt fyrirmælum um verkefnaskil má búast við niðurlækkun á einkun**
4. **Sein skil**
   - Sjá late submission policy

1 Yfirlit

Í þessu verkefni er markmiðið að smíða og hanna Python logging framework með því að nota hönnunmynstur og design principles sem við höfum séð hingað til.

Hvatinn bakvið þetta verkefni er að nemendur læri bæði að smíða og hanna hugbúnað sem og að læra að þekkja og bera kennsl á þá staði í hönnuninni og kóðanum þar sem hin helstu hönnunarmynstur eiga við og geta verið nýtt.

2 Umhverfi

Eftirfarandi kröfur eru gerðar til þess umhverfis sem er unnið við og hvernig þið ættuð að miðla því umhverfi til dæmatímakennarana

- Python 3.6+, takið fram nákvæmlega það python interpreter version sem þið eruð viss um að forritið ykkar virkar með
- Notið virtual environment til að tryggja að kerfið keyrist einnig á vélum dæmatímakennarana.

3 Structured Logging Framework (70 stig)

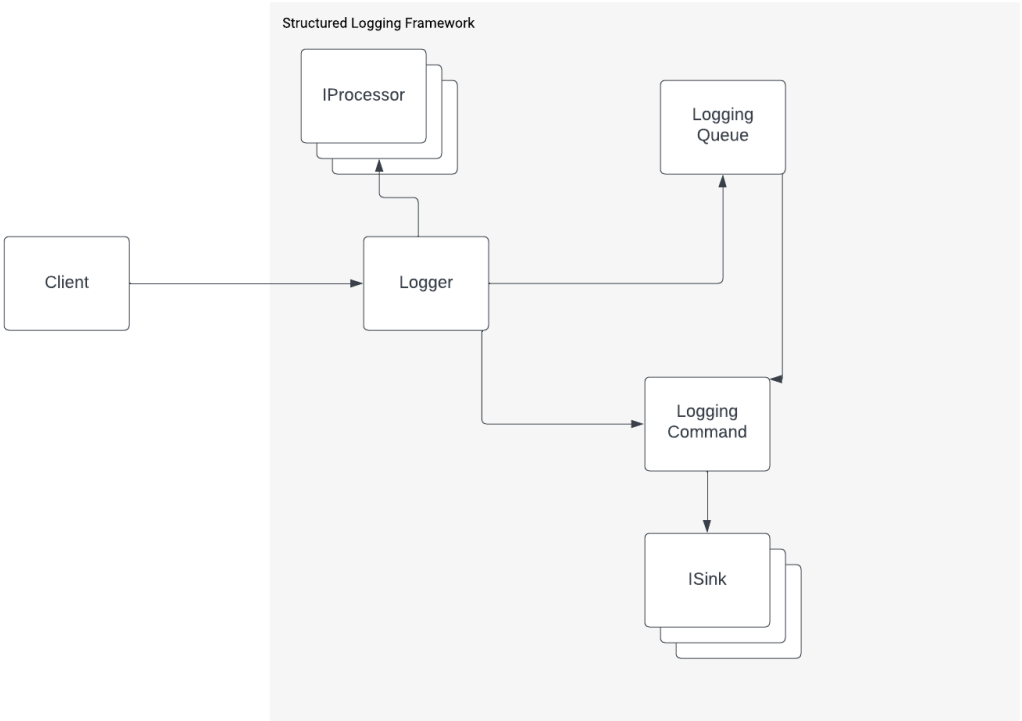
Í þessum part eigi þið að útfæra logging framework-ið. Þetta framework á að vera einfalt og extendable.

Í stuttu máli útfæriði `Logger` klasa sem hefur `log(self, **kwargs: Iterable[Any])` fall til að log-a í eitthvað tiltekið source.

logging framework-ið hefur útfærslur fyrir tvö log source (**sinks**) sem eru `ConsoleSink` sem log-ar út í console og síðan `FileSink` sem log-ar í tiltekið file. Þetta á að vera útfært þannig að það er auðvelt að bæta við fleiri sinkum án þess að gera miklar breytingar á kóða.

Einnig hefur þessi logger þann eiginleika að það er hægt að extend-a logging data með svokölluðum `processors`.

Loggerinn á síðan einnig að geta keyrt í bakgrunninum asynchronously og þannig ekki haft stór performance áhrif á notanda frameworksins.



3.1 Sinks (12.5 stig)

Í þessum part eigiði að útfæra `Sinks` þessir sinks eru þeir sem sjá um að log-a gögnin niður í eitthvað source. Passið að innri logic þessar-a sink-a er falin frá notendum þessa klasa. Notandinn ætti eingöngu að þurfa að vita að þetta er einhvers skonar sink.

3.1.1 ConsoleSink (5 stig)

Ætti að hafa fall sem tekur við `dictionary` hlut og skrifar niðurstöður þess hlutar út sem json í Console.

3.1.2 FileSink (5 stig)

Ætti að hafa fall sem tekur við `dictionary` hlut og skrifar niðurstöður þessh hlutar út sem json í skrá sem er tilgefin sem file path parameter í constructor `FileSink` klasans.

3.1.3 Hvað er mynstrið? (2.5 stig)

Útfærslan á þessum sinks er ákveðið pattern, hvað er pattern-ið?

Ath. Ég mæli með að þið prófið þessa klasa áður en þið haldið áfram (Gæti verið stekur leikur að nota pytest).

3.2 Processors (25 stig)

Í þessum lið eigi þið að útfæra Processor virknina sem er virknin sem keyrist áður en logging logic-in keyrist. Þessir processors geta þá brugðist við logging data-inu á einhvern hátt t.d. með því að bæta við það, breyta því eða kasta villu.

Þessir processors eiga að vera útfærðir með `Chain of Responsibility` pattern-inu þar sem hver processor er keyrður á eftir öðrum.

Við útfærslu á `Chain of Responsibility` þá á einnig að nota `Template Method`

3.2.1 Chain of Responsibility (10 stig)

Búið til `AbstractProcessor` sem hjúpar chain of Responsibility virknina og gerir auðvelt að búa til aðra processors með því að hjúpa *processing algorithm-an* með `Template Method`.

Processors eiga að geta bætt við virkni fyrir loggun. Þ.e.a.s að það er t.d. hægt að bæta við gögnum áður en haldið er áfram í keðjunni.

3.2.2 NullProcessor (5 stig)

Útfærið `NullProcessor` sem hefur enga virkni þ.e.a.s gerir ekkert fyrir eða eftir að keðjunni er haldið áfram. Þessi klasi verður notaður seinna af öðrum klösum og ætti því ekki að vera *notaður* í þessum lið.

3.2.3 TimestampProcessor (5 stig)

Útfærið `TimestampProcessor` sem bætir við `timestamp` sem key og tímann núna sem value í dictionary-ið.

3.2.4 EnvironmentProcessor (5 stig)

Útfærið `EnvironmentProcessor`sem bætir við `environment` key og tiltekið environment sem value. Environment sem hægt er að skrá má finna í `structured_logging/configuration/environment.py`

3.3 Logger Config Builder (10 stig)

Í `configuration` möppunninni í skeleton kóðanum má finna LoggerConfig model klasa sem skilgreinir hvernig logger-inn á að haga sér.

- **sink** → Sink property-ið tilgreinir hvaða sink logger-inn mun nota
- **processor** → Tilgreinir processing vinnuna sem á að framkvæma fyrir ingefið logging data.
- **is_async** → Segir til um hvort logging virknin keyrist í synchronously eða asynchronously þ.e.a.s í bakgrunninum eða ekki.
- **async_wait_delay_in_seconds** → Segir til um hversu lengi á að bíða í async virkninni áður en það er logað til að koma í veg fyrir óþarfa vinnslu þegar það er ekkert til að log-a.

Skeleton fyrir Builder-inn má finna undir `structured_logging/logger_creation/logger_config_builder.py`

Default sem skilast út úr þessum Builder er:

- **sink** = ConsoleSink
- **processor** = NullProcessor
- **is_async** = false
- **async_wait_delay_in_seconds** = 0

Klárið útfærslu á þessum klasa.

3.4 Queue (10 stig)

Til þess að geta gert logging virknina async þegar sú stilling er á þá munum við nota `Command Pattern`-ið ásamt processing queue til að hakka log command-in niður á sér þræði.

3.4.1 Command Queue (5 stig)

Útfærið `Queue` klasann sem má finna undir `structured_logging/command_queue`. Þessi klasi á að vinna úr lista af `Command` hlutum í endalausri lykkju. Ef listinn er tómur þá á queue-ið að bíða í `async_wait_delay_in_seconds` áður en reynt er aftur.

3.4.2 Logging Command (5 stig)

Hjúpið logging virknina í sér `LoggingCommand` hlut. Búið til þennan klasa í `structured_logging/logger` folder.

Þessi hlutur á að sjá um að log gögnin niður með því tilgefnum sink og data dictionary.

Þessi Command hlutur er sá sem Queue-ið mun vinna úr.

3.5 Logger (7.5 stig)

Útfærið nú `Logger` klasann. Beinagrind af þessum klasa má finna `structured_logging/logger/logger.py`

Log fallið gerir eftirfarandi:

1. Keyrir processing pipeline-ið út frá gefnum processor
2. Býr til LoggingCommand
3. ef `is_async` er satt á þá bætir hann því á queue-ið annars keyrir Logger hluturinn command-ið sjálfur.

3.6 Dependency Injection (5 stig)

Útfærið dependency injection í klasa undir `structured_logging/infrastructure/app_module.py`. Þessi kóði er síðan notaður í `structured_logging/logger_creation/logger_factory.py` þar sem útfærslan er gefin.

4 Client (30 stig)

Eftir að þið hafið útfært Logging Framework-ið þá er kominn tími til að nota það. Í Code skeleton-inu má einnig finna `Client` kóða sem hefur **mjög** einfalda order virkni. Það sem þið þurfið að gera er að aðlaga Logging Framework-ið að client kóðanum þannig client-inn geti farið að nota það.

Dæmi um output þegar main er keyrt væri:

Command Line

```
{
  "message": "Order started",
  "level": "info",
  "environment": "production",
  "timestamp": "08/10/2022 13:58:51"
}
{
  "message": "Payment started",
  "level": "info",
  "environment": "production",
  "timestamp": "08/10/2022 13:58:51"
}
{
  "message": "Payment finished",
  "level": "info",
  "environment": "production",
  "timestamp": "08/10/2022 13:58:51"
}
{
  "message": "Order finished",
  "level": "info",
  "environment": "production",
  "timestamp": "08/10/2022 13:58:51"
}
```

4.1 Aðlögun (15 stig)

4.1.1 Útfærsla (10 stig)

Undir `client/infrastructure/logging/I_logger.py` má finna client logger interface sem er notað í gegnum client kóðann. Aðlagið client-inn að logging framework-inu án þess að breyta tilstandandi kóða. Hafið nýja kóðann í `client/infrastructure/logging`

4.1.2 Hvað er pattern-ið? (2.5 stig)

Hvaða pattern notuðu þið til að útfæra þessa virkni?

4.1.3 Hvað er principle-ið? (2.5 stig)

Hvaða principle eru þið að aðhyllast með því að útfæra þetta svona? Útskýrið.

4.2 LoggerConfigFactory (10 stig)

Útfærið `LoggerConfigFactory` undir `client/infrastructure/logging/logging_config_factory.py` út frá þeim settings sem er gefið inn.

Ath. Í python eru functions first class citizens (og í raun útfært sem hlutir) þannig eins og stendur er þessi *factory* einfaldlega bara fall þar sem þessi virkni krefst ekkert internal state. Þar sem föll eru first class citizens í python þá getum við ennþá compose-að aðra klasa út frá þessum föllum (og getum einnig compose-að önnur föll af þessum föllum og unnið með functional programming). Þannig þótt þetta er ekki klasi þá getum við ennþá stuðlað að öllum þeim principles, patterns og clean code sem við höfum séð hingað til.

4.3 Dependency Injection (5 stig)

Vírið allan client kóðann saman með dependency injection. Útfærið dependency injection í klasa undir `client/infrastructure/dependency_injection/app_module.py`

5 Bónus (15 stig)

5.1 Bætið við virkni til að log-a hluti (5 stig)

Bætið við virkni í client logging kóðann (breytið eingöngu client logger interface-inu og kóðanum sem þið bættuð við í4.1) þannig þið getið einnig loggað heilu hlutina sem eru sendir inn.

Loggið `payment` hlutinn í `PaymentServiceStub` service.

5.2 Masking Processor (10 stig)

Í `client/infrastructure/logging` búið til `MaskingProcessor` sem tekur inn lista af lyklum til að mask-a. Þessi processor breytir síðan innihaldi dictionary-sins þannig að öll value undir þessum lyklum verða `***`.

Eftir að hafa útfært þetta, bætið processor-num í `LoggingConfig` via LoggerConfigFactory og mask-ið viðkvæmar kortaupplysingar (`card_number`, `security_code`).