Skriftlig eksamen for 1. semester datamatiker – Efterår 2017

Klasse l17dat1ae | Klasse l17dat1be | Klasse l17dat1ce

Prøvens varighed:	4 timer
Dato:	20.12.2017 kl. 8.30 til 12.30
Tilladte hjælpemidler:	
	Alle skriftlige hjælpemidler
	Det er ikke tilladt at kommunikere med andre under eksamen. Overtrædelse vil medføre bortvisning.
	IT afdelingen logger al trafik over skolens netværk, og de studerende skal aflevere deres mobiltelefoner ved prøvens start.

Eksamenssættet består af en række delopgaver. Vægtning af de enkelte opgaver fremgår af opgavebeskrivelserne.

Procenttallene for opgaverne er vejledende.

Materiale:

Du skal downloade en zip fil på Moodle (eksporteret Netbeans projekt), der indeholder en række klasser, interfaces, jUnit test samt en datafil.

Aflevering:

Du skal aflevere din besvarelse som et Netbeans projekt eksporteret som .zip-fil.

Du skal uploade din aflevering til WiseFlow.

Soundear

Virksomheden SoundEar laver støjmålere, som bruges bl.a. i institutioner og hospitaler i Danmark og andre steder i Europa.

Måske har du set det velkendte øre rundt omkring.

Støjmålerne er avancerede stykker hard- og software, men en del af deres funktionalitet kan simplificeres til følgende:

Simple sampling

Støjmåleren registrerer med korte intervaller (flere gange i minuttet) støjniveauet målt i Decibel (Db). Lydstyrken (også kaldet amplituden), gemmes sammen med en dato og et tidspunkt.

Slow sampling

Der foretages flere målinger i minuttet, men der gemmes kun en enkelt værdi for amplituden, altså en middelværdi, for hvert minut. I mange tilfælde vil denne middelværdi være tilstrækkelig, men da der kan forekomme store udsving inden for det målte minut, uden at det kan ses på middel-amplituden, registreres også en peak-værdi (det største udsving af de foretagne målinger).

Når der laves slow-sampling, vil der altid registreres både amplitude og peak (begge i Db). I modsætning til dette, vil simple-sampling altid kun registrere amplituden.

Opgaven

Det er din opgave at indlæse samples fra en given fil og oprette objekter, der overholder visse interfaces, således at data er let tilgængelig. Det er ikke meningen, at du skal skrive et fuldt færdigt projekt, men en række metoder/klasser, der overholder nogle interfaces og kan bestå nogle unit-tests.

Du skal:

1. Udfylde metoden "readFile" **(10%)**
   Metoden skal åbne filen Samples.csv, hvis navn er givet som et parameter og returnere filens indhold som en String (se String format i unit testen til readFile metoden).

2. Udfylde metoden "getSamples" **(13%)**
   Givet en String med data (i praksis, den String som readFile-metoden returnere), som er formateret korrekt, skal data konverteres til en ArrayListe af typen Sample.

3. Tilpasse metoden "getSamples" **(9%)**
   Hvis metoden tildeles data, hvor hvert data-punkt har 4 værdier, skal disse i stedet konverteres til typen SlowSample (den sidste værdi er *Peak*). – Du skal altså først skrive en klasse som implementerer dette interface (se bilag 2).

4. Tilpasse metoden "getSamples" **(7%)**
   I metoden skal der oprettes instanser af klassen Time. Såfremt Time forsynes med et parameter, som er formateret forkert, kastes en IllegalArgumentException, som er en unchecked exception. Fang denne exception og kast i stedet en TimeFormatException (klassen findes i projektet), som er checked.

5. Udfylde metoden "getHighestAmplitude" **(8%)**
   Metoden gives en liste af Sample objekter (i praksis, den liste som getSamples returnerer) og skal returnere den Sample, som har den højeste amplitude.

6. Udfylde metoden "getBiggestRise" **(11%)**
   Givet en liste af Sample objekter, returneres den sample hvor den positive forskel i amplitude set i forhold til forrige sample er den højeste. F.eks.: Hvis B's amplitude er 20 større end A's og C's er 10 højere end B's, skal B returneres.
   Hvis listen ikke eksisterer eller er tom returneres *null*. Hvis liste kun har én sample returneres den Sample.

7. Udfylde metoden "isTooLoud" **(8%)**
   Metoden returnerer true, hvis amplituden af en sample på en given liste overstiger en given grænseværdi.

8. Udfylde metoden "sortByTime" **(8%)**
   Metoden sorterer en given liste efter tidligste tidspunkt (dato ignoreres). Bemærk at Time klassen er *comparable*.

9. Udfylde metoden "sortByAmplitude" **(7%)**
   Metoden sorterer en given liste efter højeste amplitude.

10. Udfylde metoden "getLoudSamples" **(9%)**
    Givet en liste af samples og en grænseværdi, returneres en ny liste, som kun indeholder de samples, hvis amplitude overstiger grænseværdien. Samples, hvis amplitude er lig med grænseværdien, skal ikke medtages.

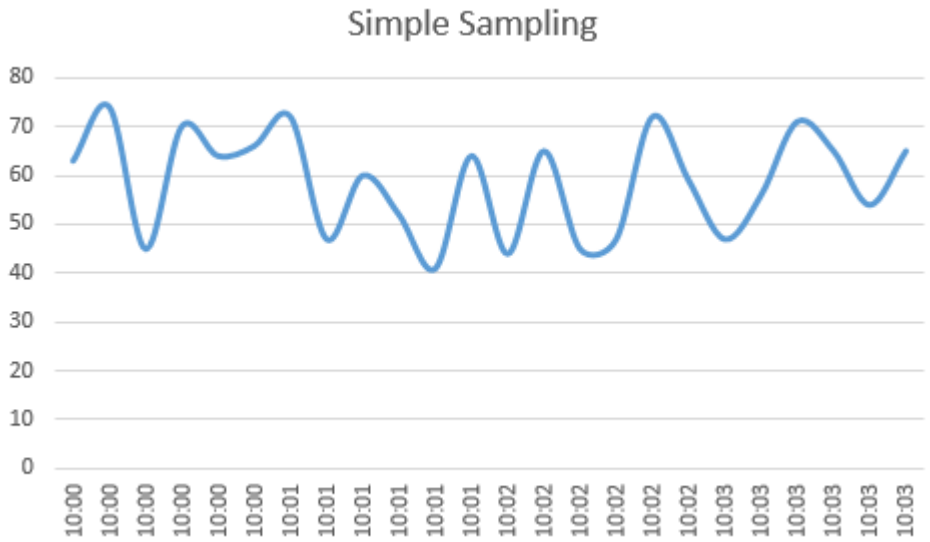
11. Udfylde metoden "getSamplesBefore" **(10%)**
    Givet en liste af samples og en grænseværdi, returneres en samling af alle de samples som er registreret før et givet tidspunkt (dato ignoreres). Bemærk, at Time er *comparable.*

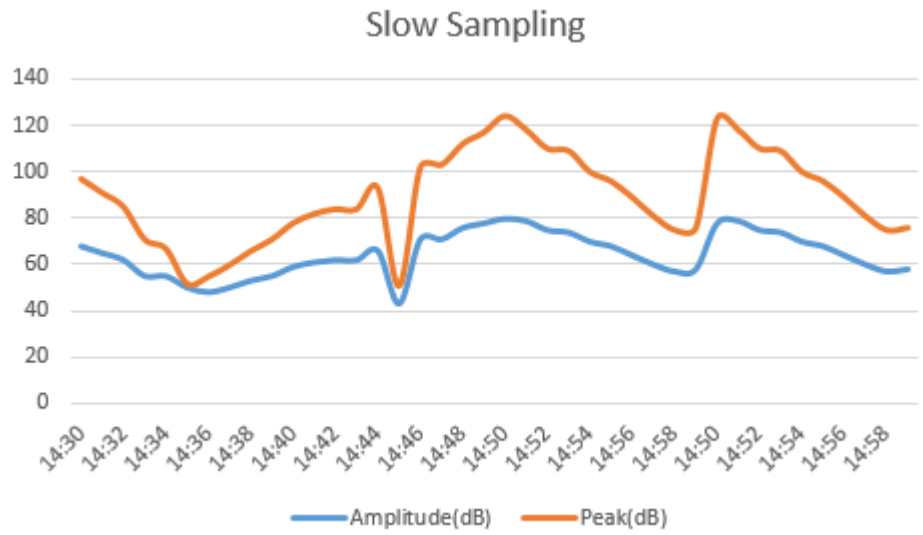
Vurdering

Bemærk, at det ikke er nok at implementere interfaces uden omtanke – den specificerede logik i opgavebeskrivelserne skal følges.

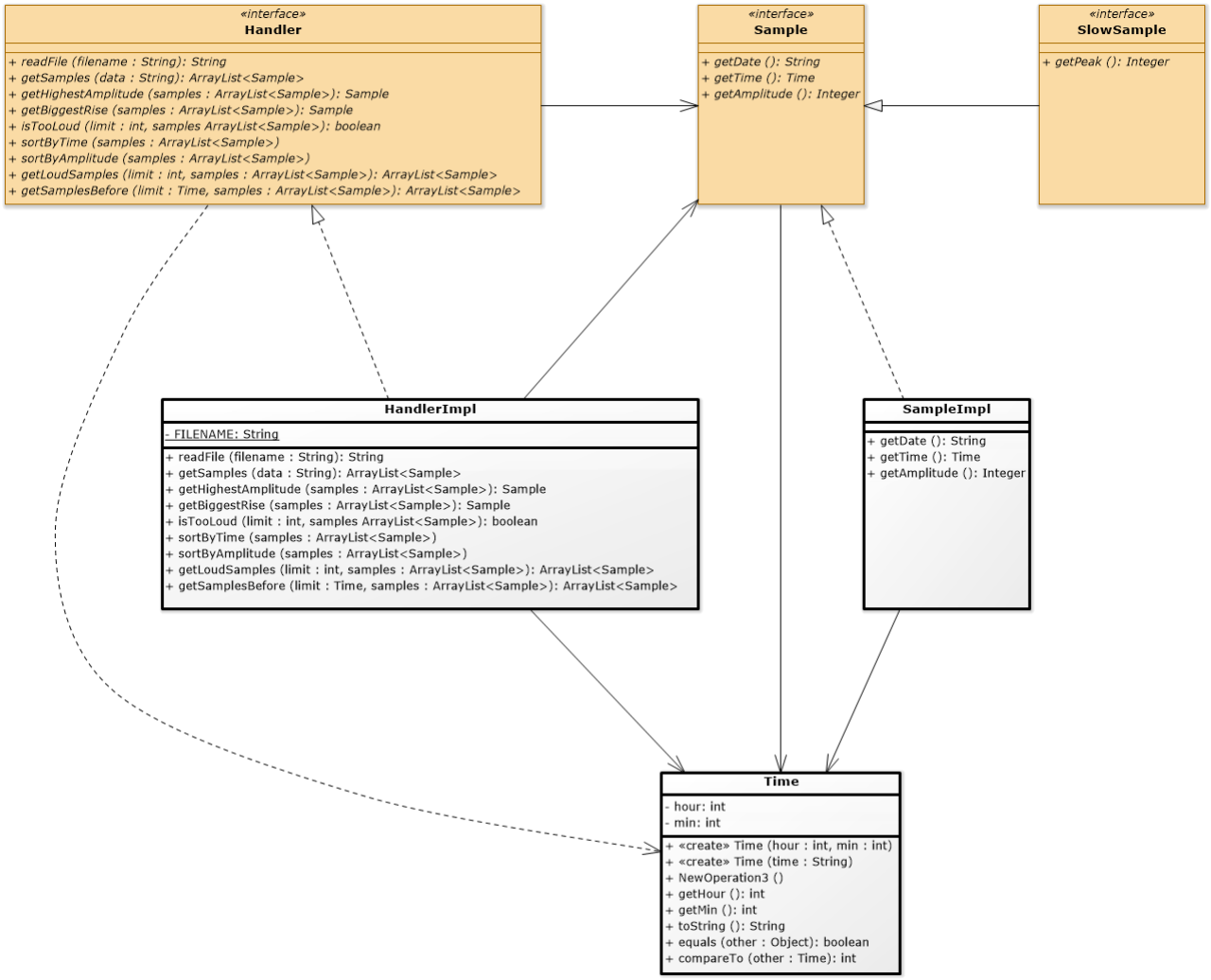
De udleverede tests er beregnet til at give dig en ide om, hvorvidt du har løst opgaven korrekt. Du kan forvente, at der vil blive kørt yderligere tests i forbindelse med vurderingen af din besvarelse. Du skal altså løse opgaverne generelt og ikke kun for de i testen given værdier. Du kan dog regne med, at andre test vil have nogenlunde samme udformning og værdier i samme størrelsesorden.

Bilag 1 – Sample eksempel





Bilag 2 – Klasse diagram



Diagrammet viser sammenhængen mellem klasser og interfaces. Bemærk at der er forskellige relationer.

Diagrammet udtrykker, at klassen `HandlerImpl` implementerer interfacet `Handler`. `HandlerImpl` kender desuden til `Time` og `Sample` (via inputparametre eller returtype).

Klassen `SampleImpl` implementerer interfacet `Sample`, som også har en reference til `Time` (som attribut). Interfacet `SlowSample` arver fra `Sample`.