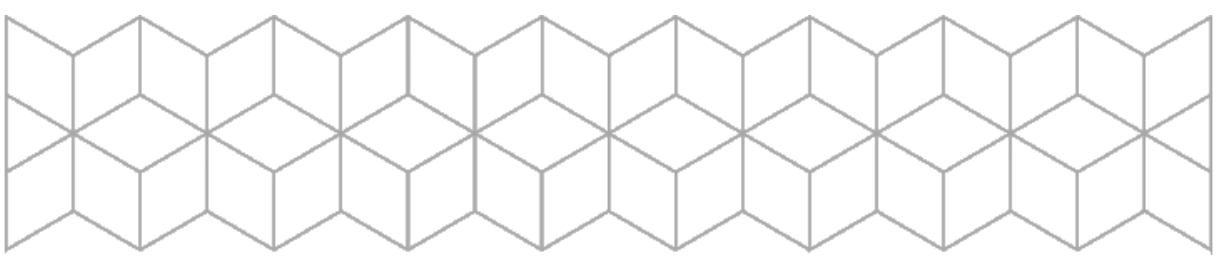
Høgskolen i Østfold

SENSORVEILEDNING

Emnekode:	ITF10611
Emnenavn:	**Objektorientert programmering**
Eksamensform:	**Digital eksamen**
Dato:	06.05.2019
Faglærer(e):	Lars Emil Knudsen
Eventuelt:	



Eksamen består av tre hoveddeler; del 1 (20%) , del 2 (20%) og del 3 (60%).

Del 1 består av 20 spørsmål. Oppgavene består av teorioppgaver i flersvarsform med fire alternativer. Det er ett alternativ som er korrekt for hver oppgave. Riktig svar gir 3 poeng, mens feil svar gir -1 poeng.

Del 2 tar for seg analyse og forståelse av Javakode og dens oppbygning, oppgavene er i flersvarsform. Det er ett alternativ som er korrekt for hver oppgave. Riktig svar gir 3 poeng, mens feil svar gir -1 poeng.

Del 3 består av konkrete kodeoppgaver i et «prosjekt». Her vil kandidaten få vist sin evne til å sette seg inn i en ukjent problemstilling og lage konkrete kodeløsninger til denne.

Evaluering

Sensor står fritt til å tilpasse poeng tilsvarende en helhetsvurdering. F.eks. vil et scenarie hvor kandidaten svarer korrekt på 1 oppgave, og feil på 3 oppgaver i del 2 resultere i 0 poeng, det anbefales derfor å gjøre egne tilpasninger i forhold til dette.

Hvis det er ønskelig og konvertere oppgavene til en konkret karakter i forhold til hvor mange prosent man har riktig kan følgende karaktergrenser benyttes:

A	90
B	80
C	60
D	50
E	40
F	0

Det skal vektlegges en helhetlig vurdering. Det vil f.eks. si at en kandidat som har alt riktig i del 1 og del 2 (som da tilsvarer 40% og totalt sett en E), likevel vil kunne få en F hvis del 3 tilsvarer en F.

Fasit og løsningsforslag for del 3 ligger vedlagt. Det er verdt og merke seg at dette er et *forslag*. Det vil si at det ikke er nødvendig at kandidaten har et svar helt i henhold til denne for å få full uttelling. Det bør gis poeng selv om bare deler av logikken er korrekt eller det er mindre skrive/kodefeil. Andre elementer kan også betegnes som en «bonus» og ikke nødvendig for å få full uttelling. Ett slikt eksempel er bruk av konstanter for de forskjellige grensene definert i oppgaven slik som:

```
private static final int MAX_KANALER = 24;
```

Karakterbeskrivelse

A	Fremragende	Fremragende prestasjon som klart utmerker seg. Kandidaten viser svært god vurderingsevne og stor grad av selvstendighet.
B	Meget god	Meget god prestasjon. Kandidaten viser meget god vurderingsevne og selvstendighet.
C	God	Jevnt god prestasjon som er tilfredsstillende på de fleste områder. Kandidaten viser god vurderingsevne og selvstendighet på de viktigste områdene.
D	Nokså god	En akseptabel prestasjon med noen vesentlige mangler. Kandidaten viser en viss grad av vurderingsevne og selvstendighet.
E	Tilstrekkelig	Prestasjonen tilfredsstiller minimumskravene, men heller ikke mer. Kandidaten viser liten vurderingsevne og selvstendighet.
F	Ikke bestått	Prestasjon som ikke tilfredsstiller de faglige minimumskravene. Kandidaten viser både manglende vurderingsevne og selvstendighet.

Vedlegg: OOP 2019 – Løsningsforslag

Oppgave 1 (20%) – Flervalgspørsmål

1.1: C
1.2: B
1.3: D
1.4: B
1.5: C
1.6: A
1.7: C
1.8: A
1.9: D
1.10: A
1.11: B
1.12: B
1.13: C
1.14: C
1.15: D
1.16: A
1.17: D
1.18: B
1.19: B
1.20: C

Oppgave 2 (20%) - Kodeforståelse

2.1. D
2.2. A
2.3. A
2.4. D

Oppgave 3 (20%) – Programmering

Oppgave 3.1

```
public class Lyd {
    private String navn;
    private int varighet, kanaler, bitPerSample;
    private long samplingsrate;
    private LydSamples lydSamples;

    private static final int MIN_KANALER = 1;
    private static final int MAX_KANALER = 24;

    public Lyd(String navn, int varighet, int kanaler,
               long samplingsrate, int bit, LydSamples lydSamples) {
        if (verifiserKanaler(kanaler))
            this.kanaler = kanaler;
        else
            throw new IllegalArgumentException(kanaler + "utenfor "
                                + MIN_KANALER + " og "
                                + MAX_KANALER + " kanaler");

        this.navn = navn;
        this.varighet = varighet;
        this.samplingsrate = samplingsrate;
        this.bitPerSample = bit;
        this.lydSamples = lydSamples;
    }

    private boolean verifiserKanaler(int kanaler) {
        return MIN_KANALER <= kanaler && kanaler <= MAX_KANALER;
    }

    public String getNavn() {
        return navn;
    }

    public int getVarighet() {
        return varighet;
    }

    public int getBitPerSample() {
        return bitPerSample;
    }

    public int getKanaler() {
        return kanaler;
    }

    public long getSamplingsrate() {
        return samplingsrate;
    }

    public int getBit() {
        return bitPerSample;
    }

    public LydSamples getLydSamples() {
        return lydSamples;
```

```
    }
}
```

Oppgave 3.2

```
private static final int MIN_KANALER_CD = 2;
private static final int MIN_SAMPLINGSRATE_CD = 44100;
private static final int MIN_BITPERSAMPLE_CD = 16;

public void spill() {
    // implementeres av andre
}

public long getSize() {
    long bytePerSample = (bitPerSample+7) / 8;

    return kanaler * samplingsrate * bytePerSample * varighet;
}

public boolean cdKvalitet() {
    return MIN_KANALER_CD <= kanaler &&
           MIN_SAMPLINGSRATE_CD <= samplingsrate &&
           MIN_BITPERSAMPLE_CD <= bitPerSample;
}
```

Oppgave 3.3

```
public abstract class Spillbar {
    protected String navn;
    protected int varighet;

    public Spillbar(String navn, int varighet) {
        this.navn = navn;
        this.varighet = varighet;
    }

    public abstract void spill();

    public abstract long getSize();

    public String getNavn() {
        return navn;
    }

    public int getVarighet() {
        return varighet;
    }
}

// Den må extende spillbar, kalle superkonstruktøren. Spill() og
getSize() bør annoteres med @Override
public class Lyd extends Spillbar{
    private int kanaler, bitPerSample, samplingsrate;
    private LydSamples lydSamples;

    private static final int MIN_KANALER = 1;
    private static final int MAX_KANALER = 24;
    private static final int MIN_KANALER_CD = 2;
    private static final int MIN_SAMPLINGSRATE_CD = 44100;
    private static final int MIN_BITPERSAMPLE_CD = 16;

    public Lyd(String navn, int varighet, int kanaler,
               int samplingsrate, int bit, LydSamples lydSamples){
        super(navn, varighet);

        if (verifiserKanaler(kanaler))
            this.kanaler = kanaler;
        else
            throw new IllegalArgumentException();

        this.samplingsrate = samplingsrate;
        this.bitPerSample = bit;
        this.lydSamples = lydSamples;
    }

    private boolean verifiserKanaler(int kanaler) {
        return MIN_KANALER <= kanaler && kanaler <= MAX_KANALER;
    }

    @Override
    public void spill() {
        // implementeres av andre
    }
```

```
    @Override
    public long getSize() {
        long bytePerSample = (bitPerSample+7) / 8;

        return kanaler * samplingsrate * bytePerSample * varighet;
    }

    public boolean cdKvalitet() {
        return MIN_KANALER_CD <= kanaler &&
               MIN_SAMPLINGSRATE_CD <= samplingsrate &&
               MIN_BITPERSAMPLE_CD <= bitPerSample;
    }

    public int getKanaler() {
        return kanaler;
    }

    public long getSamplingsrate() {
        return samplingsrate;
    }

    public int getBit() {
        return bitPerSample;
    }

    public LydSamples getLydSamples() {
        return lydSamples;
    }
}
```

Oppgave 3.4

```
public class Video extends Spillbar {
    private Lyd lyd;
    private BildeSekvens bildeSekvens;

    public Video(String navn, Lyd lyd, BildeSekvens bildeSekvens) {
         // Det antas her at bildeSekvens har en varighet
         super(navn,
              Math.max(lyd.getVarighet(),
     bildeSekvens.getVarighet()));
        this.lyd = lyd;
        this.bildeSekvens = bildeSekvens;

    }

    @Override
    public void spill() {
       // Implementeres av andre
    }

    @Override
    public long getSize() {
        // Det antas her at bildeSekvens har en størrelse
        return lyd.getSize() + bildeSekvens.getSize();
    }

    // Vanlige get-metoder bør være med
}
```

Oppgave 3.5

```
public abstract class Spillbar implements Comparable<Spillbar> {
    //...
    @Override
    public int compareTo(Spillbar annenSpillbar) {
        return navn.compareTo(annenSpillbar.getNavn());
    }
}
```

Oppgave 3.6

Kan override toString() i Lyd og Video-klassene på «vanlig» måte. Alternativt kan man lage en abstrakt type() i Spillbar og override toString i Spillbar. Man bør ikke gjøre det i Spillbar ved bruk av instanceOf, da dette innebærer at superklassen vet om sine «barn», noe den ikke bør gjøre.

```
public abstract class Spillbar {
    public abstract String type();

    @Override
    public String toString() {
        return type() + navn + " " + varighet;
    }
}
```

Så må denne implementeres i Lyd og Video:

```
@Override
public String type() {
    return "Lyd - ";
}
```

Oppgave 3.7

```
ObservableList<Spillbar> oList =
FXCollections.observableArrayList();

oList.add(new Lyd("Philter: We Move like wolves", 190, 16, 44100,
                        24, new LydSamples()));
oList.add(new Video("Jurassic World", new Lyd("JW", 7440),
                              new BildeSekvens("JW", 7440)));

Collections.sort(oList);
mediaListe.setItems(oList);
```

Oppgave 3.8

```
if (nySpillbar != null) {
    statusTekst.setText("Spiller nå: " + nySpillbar.getNavn());

    // Både 9 og 10 vil være OK
    progressSlider.setMajorTickUnit(nySpillbar.getVarighet() / 9f);
    progressSlider.setMax(nySpillbar.getVarighet());
    progressSlider.setValue(nySpillbar.getVarighet() * 0.3f);
    // Bonus hvis kandidaten har med denne:
    nySpillbar.spill();
}
```