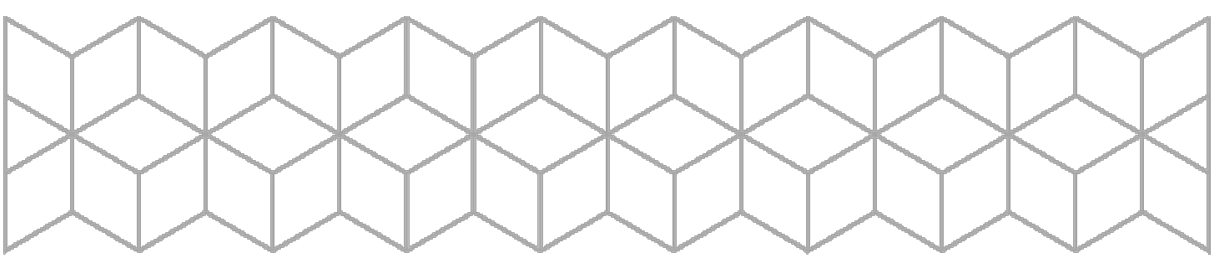
Høgskolen i Østfold

# SENSORVEILEDNING

| | |
|---|---|
| **Emnekode:** | ITF10619 |
| **Emnenavn:** | **Programmering 2** |
| **Eksamensform:** | **Digital eksamen** |
| **Dato:** | 09.05.2023 |
| **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 1 poeng, mens feil svar gir 0 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 1 poeng, mens feil svar gir 0 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.

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.

Løsningforslag 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 MIN_ANTALL_REKKER = 1;
```

## 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 Programmering 2 – 2023 – Løsningsforslag

## Del 3 (60%) – Programmering

## Oppgave 3.1

```
1 inheritor  Lars Emil +1 *
public class TallRekke {
    5 usages
    private Set<Integer> tallRekke = new TreeSet<>();
    1 usage
    private static final int ANTALL_TALL = 7;
    2 usages
    protected static final int MIN_TALL = 1;
    2 usages
    protected static final int MAKS_TALL = 34;

    Lars Emil
    public TallRekke() {
        while (tallRekke.size() < ANTALL_TALL) {
            int tilfeldigTall = (int) (Math.random() * MAKS_TALL + MIN_TALL);
            tallRekke.add(tilfeldigTall);
        }
    }

    new *
    public TallRekke(Set<Integer> tall) { this.tallRekke = tall; }

    6 usages  Lars Emil
    public Set<Integer> getTallRekke() { return tallRekke; }

    Lars Emil
    @Override
    public String toString() { return tallRekke.toString(); }
}
```

## Oppgave 3.2

```java
Lars Emil +1
public class VinnerRekke extends TallRekke {
    5 usages
    private int ekstraNummer = 0;

    Lars Emil +1
    public VinnerRekke() {
        super();

        while (ekstraNummer == 0 || getTallRekke().contains(ekstraNummer)) {
            ekstraNummer = (int) (Math.random() * MAKS_TALL + MIN_TALL);
        }
    }

    6 usages  Lars Emil +1
    public int antallKorrekteTall(TallRekke annenTallRekke) {
        int antallRiktige = 0;

        for (int tall : annenTallRekke.getTallRekke()) {
            if (getTallRekke().contains(tall))
                antallRiktige++;
        }

        return antallRiktige;
    }

    2 usages  Lars Emil
    public boolean harEkstraNummer(TallRekke tallRekke) {
        return tallRekke.getTallRekke().contains(ekstraNummer);
    }

    no usages  Lars Emil
    public int getEkstraNummer() { return ekstraNummer; }
}
```

# Oppgave 3.3

```java
Lars Emil +1
public class Kupong {
    2 usages
    private static final int MIN_ANTALL_REKKER = 1;
    2 usages
    private static final int MAX_ANTALL_REKKER = 10;
    2 usages
    private final int id;
    1 usage
    private static int idTeller = 0;
    3 usages
    private ArrayList<TallRekke> tallRekker;

    Lars Emil +1
    public Kupong(int antallRekker) {
        if (antallRekker < MIN_ANTALL_REKKER || antallRekker > MAX_ANTALL_REKKER)
            throw new IllegalArgumentException("Antall rekker må være mellom " + MIN_ANTALL_REKKER + " og " + MAX_ANTALL_REKKER);

        id = idTeller++;
        tallRekker = new ArrayList<>();

        for (int i = 0; i < antallRekker; i++) {
            tallRekker.add(new TallRekke());
        }
    }

    1 usage  Lars Emil
    public int getId() { return id; }

    6 usages  Lars Emil
    public ArrayList<TallRekke> getTallRekker() { return tallRekker; }
}
```

## Oppfgave 3.4

```
28          app.get( path: "/api/vinnerrekke", ctx -> {
29              ctx.json(vinnerRekke);
30          });
31
32          app.get( path: "/api/kupong/{kupong-id}", ctx -> {
33              int kupongid = Integer.parseInt(ctx.pathParam( s: "kupong-id"));
34
35              for (Kupong kupong : alleKuponger) {
36                  if (kupong.getId() == kupongid) {
37                      ctx.json(kupong);
38                      return;
39                  }
40              }
41
42              ctx.status(404).result( resultString: "Kupong ikke funnet");
43          });
```

## Oppgave 3.5

```java
public static List<Kupong> kalkulerVinnerKuponger(List<Kupong> alleSpilteKuponger, VinnerRekke vinnerRekke) {
    // Oppgave 3.5
    List<Kupong> vinnerKuponger = new ArrayList<>();

    for (Kupong kupong : alleSpilteKuponger) {
        for (TallRekke tallRekke : kupong.getTallRekker()) {
            if (vinnerRekke.antallKorrekteTall(tallRekke) >= 4) {
                vinnerKuponger.add(kupong);
                break;
            }
        }
    }

    return vinnerKuponger;
}
```

## Oppgave 3.6

```java
public static int kalkulerPremieForHverVinnerIPremieKategori(PremieKategori premieKategori,
                                                             int antallVinnereIPremieKategori,
                                                             int totalPremiePott) {
    // Oppgave 3.6
    if (antallVinnereIPremieKategori == 0)
        return 0;

    double prosentAndel = 100;

    switch (premieKategori) {
        case SYV -> prosentAndel = 40;
        case SEKS_PLUSS_EN -> prosentAndel = 5;
        case SEKS -> prosentAndel = 5;
        case FEM -> prosentAndel = 20;
        case FIRE -> prosentAndel = 30;
    }

    return (int) (totalPremiePott * (prosentAndel / 100) / antallVinnereIPremieKategori);
}
```