



KANDIDAT  
10009

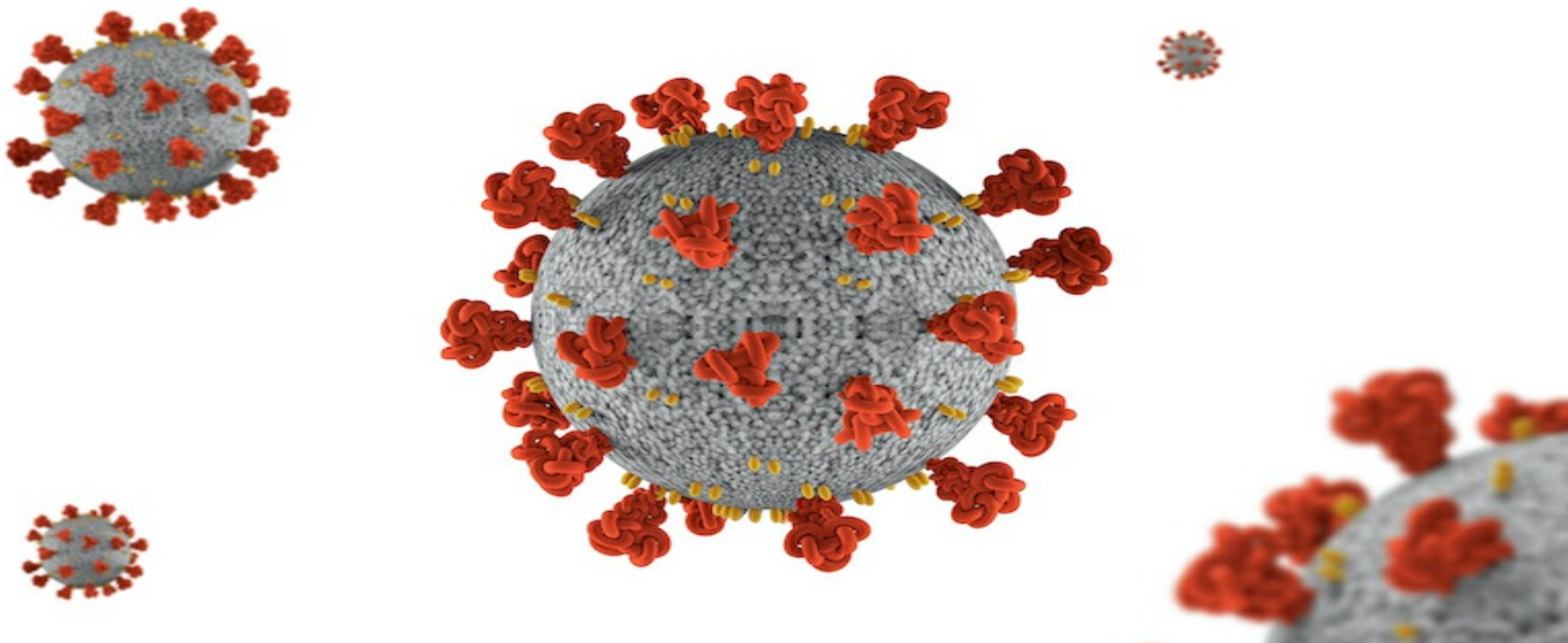
PRØVE  
IDATG1001 1 Programmering 1

|                |                  |
|----------------|------------------|
| Emnekode       | IDATG1001        |
| Vurderingsform | Hjemmeeksamen    |
| Starttid       | 14.12.2020 08:00 |
| Sluttid        | 14.12.2020 12:00 |
| Sensurfrist    | 14.01.2021 22:59 |
| PDF opprettet  | 13.02.2021 14:27 |

| Forside                   |                                  |               |
|---------------------------|----------------------------------|---------------|
| Oppgave                   | Tittel                           | Oppgavetype   |
| i                         | Forside                          | Dokument      |
| Oppgave 1 (ca 10%)        |                                  |               |
| Oppgave                   | Tittel                           | Oppgavetype   |
| 1(a)                      | Oppgave 1 a) - Klassediagram     | Filopplasting |
| 1(b)                      | Oppgave 1 b) - Aktivitetsdiagram | Filopplasting |
| 1(c)                      | Oppgave 1 c)                     | Langsvar      |
| Oppgave 2 (ca 15%)        |                                  |               |
| Oppgave                   | Tittel                           | Oppgavetype   |
| 2(a)                      | Oppgave 2 a)                     | Langsvar      |
| 2(b)                      | Oppgave 2 b)                     | Langsvar      |
| 2(c)                      | Oppgave 2 c)                     | Langsvar      |
| 2(d)                      | Oppgave 2 d)                     | Langsvar      |
| Oppgave 3 (ca 40%)        |                                  |               |
| Oppgave                   | Tittel                           | Oppgavetype   |
| 3(a)                      | Oppgave 3 a)                     | Programmering |
| 3(b)                      | Oppgave 3 b)                     | Langsvar      |
| 3(c)                      | Oppgave 3 c)                     | Programmering |
| Oppgave 4 (ca 35%)        |                                  |               |
| Oppgave                   | Tittel                           | Oppgavetype   |
| 4(a)                      | Oppgave 4 a)                     | Programmering |
| 4(b)                      | Oppgave 4 b)                     | Langsvar      |
| Opplasting av Prosjektfil |                                  |               |

| Oppgave | Tittel                                         | Oppgavetype   |
|---------|------------------------------------------------|---------------|
| 5       | Opplasting av prosjektfil                      | Filopplasting |
| Vedlegg |                                                |               |
| Oppgave | Tittel                                         | Oppgavetype   |
| i       | Vedlegg 1 - Klientprogram med JOptionPane      | Dokument      |
| i       | Vedlegg 2 - Klientprogram med tekstbasert meny | Dokument      |

1



## Oppgave 1 (ca 10%)

# Problembeskrivelse

## Covid19 register

Den 31 desember 2019 rapporterte Helsedepartementet i Wuhan i Kina om et influensautbrudd av ukjent opprinnelse. 44 pasienter ble innrapportert med influensasymptomer. 11 av disse var kritisk syke. Dette ble starten på en av de værste pandemiene i verden i moderne historie. Sykdommen fikk senere navnet COVID-19, og man kom frem til at sykdommen var forårsaket av et (av mange) coronavirus som mest sannsynlig stammer fra en bestemt rase av flaggermus i Kina. Dette viruset fikk navnet "SARS-CoV-2".

Utbredelsen av COVID-19 loggføres nå over hele verden.

## Innledning

Les denne kravspesifikasjonen grundig. Gjør deretter oppgavene, som vil lede deg igjennom implementasjonen av dette prosjektet.

## Kravspesifikasjon

Det skal utvikles et system for å registrere underlagsdata for til bruk for å utarbeide ulike statistikker. Hver registrering skal inneholde følgende informasjon:

- Dato for registrering
- Landet registreringen gjøres i (engelsk: country) - som tekst
- Antall nye smittede (engelsk: number of infected) - positivt heltall
- Antall nye døde (engelsk: number of deaths) - positivt heltall

I tabellen under finner du noen eksempler du kan bruke i din løsning:

| Dato       | Land  | Smittede | Døde |
|------------|-------|----------|------|
| 19.01.2020 | CHINA | 136      | 1    |
| 05.02.2020 | CHINA | 3872     | 66   |
| 07.03.2020 | NORGE | 3        | 0    |
| 09.03.2020 | USA   | 259      | 4    |
| 09.03.202  | CHINA | 45       | 23   |

|            |       |       |      |
|------------|-------|-------|------|
| 22.03.2020 | NORGE | 240   | 8    |
| 24.03.2020 | USA   | 20341 | 119  |
| 25.03.2020 | CHINA | 28    | 4    |
| 06.04.2020 | NORGE | 110   | 3    |
| 10.04.2020 | USA   | 30859 | 2087 |
| 10.04.2020 | CHINA | 55    | 1    |

## Funksjonalitet

Applikasjonen skal ha følgende funksjonalitet:  
Brukeren må kunne

- registrere/legge inn COVID-19 tilfeller, hver registrering er pr dag.
- skrive ut liste over alle registreringer
- søke etter en registrering basert på dato
- søke etter registreringer *etter* en gitt dato
- regne ut samlet antall døde for et gitt land

## Vurderingskriterier

Din besvarelse vil bli vurdert basert på følgende kriterier:

- Om du har fulgt (og holdt deg til) kravspesifikasjonen
- Om du har fulgt prinsippene for god design (coupling, cohesion, responsibility driven design osv)
- Om du har valgt gode, selvforklarende navn på klasser, metoder og variabler/felt/parametre
- Om du har dokumentert koden din godt (Javadoc)
- Om du har god og fornuftig samhandling med bruker (et godt brukergrensesnitt)

I de påfølgende deloppgavene skal du gradvis utvikle applikasjonen som skal brukes for å håndtere registreringen av COVID-tilfeller.

(a) **Oppgave 1 a) - Klassediagram**

Denne oppgaven skal du løse ved å tegne på papir eller ved bruk av programvare for tegning. Tegningen skanner du/lager du PDF eller JPG-fil av og laster opp i oppgaven.

**Oppgave 1 a) - Klassediagram**

Les problembeskrivelse/kravspesifikasjonen og lag/tegn ett **klassediagram** for alle klassene du mener bør utvikles.

Vis i klassediagrammet hvilke **relasjoner** som finnes mellom klassene (hvem kjenner til hvem) og om det er **kardinalitet** (en-til-en, mange-til-en osv) knyttet til relasjonen.

Din fil ble lastet opp og lagret i besvarelsen din.

Last ned

Fjern


Erstatt

|                       |                   |
|-----------------------|-------------------|
| Filnavn:              | Klassediagram.jpg |
| Filtype:              | image/jpeg        |
| Filstørrelse:         | 96.45 KB          |
| Opplastingstidspunkt: | 14.12.2020 11:29  |
| Status:               | Lagret            |

(b) Oppgave 1 b) - Aktivitetsdiagram

Oppgave 1 b) - Aktivitetsdiagram

Fra kravspesifikasjonen, tegn et **aktivitetsdiagram** som viser programflyten sett fra sluttbruker.



Din fil ble lastet opp og lagret i besvarelsen din.

Last ned

Fjern

Erstatt

|                       |                       |
|-----------------------|-----------------------|
| Filnavn:              | Aktivitetsdiagram.jpg |
| Filtype:              | image/jpeg            |
| Filstørrelse:         | 1.96 MB               |
| Opplastingstidspunkt: | 14.12.2020 12:09      |
| Status:               | Lagret                |

(c) **Oppgave 1 c)**

Beskriv hvordan du gikk frem for å identifisere kandidater til klassene. Beskriv også hva som er **oppgaven/rollen/ansvaret** til hver av klassene (tenk Javadoc av klassen).

**Skriv ditt svar her**

For "CovidLocationStats" får man vite i oppgaven: Lag en klasse som representerer en COVID registrering. Man får så oppgitt i problembeskrivelsen:

Hver registrering skal inneholde følgende informasjon:

- Dato for registrering
- Landet registreringen gjøres i (engelsk: country) - som tekst
- Antall nye smittede (engelsk: number of infected) - positivt heltall
- Antall nye døde (engelsk: number of deaths) - positivt heltall

Ut ifra dette manglet det da kun å gi riktig datatype til de forskjellige kandidatene.

Videre må man ha et register som inneholder alle disse covid-statsene. Denne må hvertfall inneholde en HashSet eller ArrayList. Alle klassene inneholder kun metoder/funksjoner som er relevante til deres klasse. Et eksempel på dette er for eksempel at koden for å utføre operasjonen for å legge alle covid-entries i en liste, ikke ligger i App-klassen som selv har hovedfokus på å kun inneholde brukergrensesnittet. I stedet ligger koden for denne operasjonen i samme klasse som hvor selve registeret i form av en HashSet ligger. Dette er kjent som high cohesion.

Har så laget en klasse kalt App som fungerer som en meny med informasjon for brukeren. Denne styres ved hjelp av en InputHandler, som sjekker om input i App-klassen er gyldig eller ikke.

Appen bruker ENUMS, en scanner, en switch og inputHandleren.



2



## Oppgave 2 (ca 15%)

En viktig ferdighet som programvareutvikler er å finne frem i dokumentasjonen av biblioteker og rammeverk som kan være nyttig å bruke i din løsning.

I denne oppgaven skal du slå opp i dokumentasjonen for **JDK 1.8 (API)** og se nærmere på klassen **LocalDate**.

Du må selv finne frem til riktig URL for dokumentasjon.

(a) **Oppgave 2 a)**

I hvilken **pakke** finner du klassen **LocalDate** ?  
Hva må du skrive i koden din for å fortelle at du kommer til å bruke akkurat denne klassen ?

Skriv ditt svar her...

```
java.lang.Object
sub: java.time.LocalDate

import java.time.LocalDate;
```

(b) **Oppgave 2 b)**

Hvilken metode i klassen vil du bruke for å lage en instans av LocalDate som representerer **dagens dato** ?

Skriv ditt svar her...

```
LocalDate.now()
```

**(c) Oppgave 2 c)**

Klassen **LocalDate** har en metode for å sjekke om en dato er tidligere enn en annen dato. Hva heter denne? Skriv **signaturen** til denne metoden i svarfeltet under.

**Skriv ditt svar her**

## LocalDate.isBefore()

Metoden heter isBefore og tar inn parameter av type ChronoLocalDate other

isBefore(ChronoLocalDate other)

**(d) Oppgave 2 d)**

LocalDate-klassen har en metode med følgende signatur:

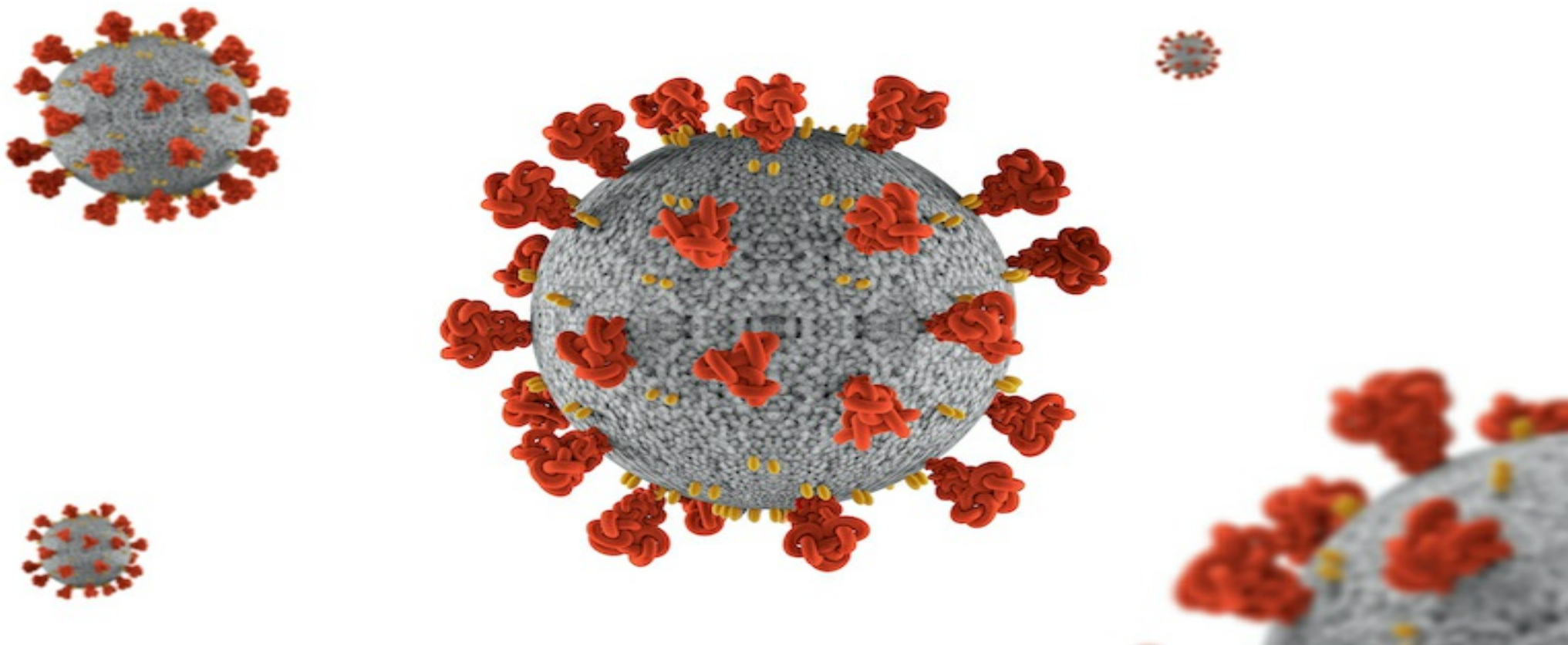
*LocalDate parse(CharSequence text)*

## Hva gjør denne metoden?

**Skriv ditt svar her...**

Får en instanse av `LocalDate` fra en tekststreng som 2007-12-03.

3



### Oppgave 3 (ca 40%)

Gjennom resten av oppgavene skal du implementere applikasjonen.  
Her kommer kravspesifikasjonen en gang til:

#### Kravspesifikasjon

Det skal utvikles et system for å registrere underlagsdata for til bruk for å utarbeide ulike statistikker.  
Hver registrering skal inneholde følgende informasjon:

- Dato for registrering
- Landet registreringen gjøres i (engelsk: country) - som tekst
- Antall nye smittede (engelsk: number of infected) - positivt heltall
- Antall nye døde (engelsk: number of deaths) - positivt heltall

I tabellen under finner du noen eksempler du kan bruke i din løsning:

| Dato       | Land  | Smittede | Døde |
|------------|-------|----------|------|
| 19.01.2020 | CHINA | 136      | 1    |
| 05.02.2020 | CHINA | 3872     | 66   |
| 07.03.2020 | NORGE | 3        | 0    |
| 09.03.2020 | USA   | 259      | 4    |
| 09.03.202  | CHINA | 45       | 23   |
| 22.03.2020 | NORGE | 240      | 8    |
| 24.03.2020 | USA   | 20341    | 119  |
| 25.03.2020 | CHINA | 28       | 4    |
| 06.04.2020 | NORGE | 110      | 3    |
| 10.04.2020 | USA   | 30859    | 2087 |
| 10.04.2020 | CHINA | 55       | 1    |

#### Funksjonalitet

Applikasjonen skal ha følgende funksjonalitet:  
Brukeren må kunne

- registrere/legge inn COVID-19 tilfeller, hver registrering er pr dag.
- skrive ut liste over alle registreringer
- søke etter en registrering basert på dato
- søke etter registreringer *etter* en gitt dato
- regne ut samlet antall døde for et gitt land

## Vurderingskriterier

Din besvarelse vil bli vurdert basert på følgende kriterier:

- Om du har fulgt (og holdt deg til) kravspesifikasjonen
- Om du har fulgt prinsippene for god design (coupling, cohesion, responsibility driven design osv)
- Om du har valgt gode, selvforklarende navn på klasser, metoder og variabler/felt/parametre
- Om du har dokumentert koden din godt (Javadoc)
- Om du har god og fornuftig samhandling med bruker (et godt brukergrensesnitt)

I de påfølgende deloppgavene skal du gradvis utvikle applikasjonen som skal brukes for å håndtere registreringen av COVID-tilfeller.

### (a) Oppgave 3 a)

Du skal nå implementere løsningen.

Bruk ditt foretrukne IDE (BlueJ, IntelliJ, Netbeans osv). Opprett et helt nytt prosjekt på din harddisk.

Lag en klasse som representerer en COVID registrering (engelsk: covid location stats):

1. Gi klassen et fornuftig og forklarende navn.
2. Definere fornuftige felt av relevante datatyper (for dato kan du for eksempel bruke klassen **LocalDate** fra oppgave 2).
3. Implementer **konstruktøren** med nødvendige parametre
4. Implementer **aksessor-metoder** for klassen.
5. Implementer **mutator-metoder** for klassen (de du mener er nødvendige/nyttige).
6. Dokumenter klassen og metodene (Javadoc)

Kopier koden for klassen fra din IDE inn i svarfeltet under.

```
1 package me.ntnu.candidate.exam;
2
3 import java.time.LocalDate;
4
5 /**
6  * This class is used to describe some covid location stats.
7  * @Author: 10009
8  */
9 public class CovidLocationStats {
10
11     private LocalDate localDate;
12     private String country;
13     private int numberOfInfected;
14     private int numberOfDeaths;
15
16     /**
17      * This is a constructor that takes 4 parameters
18      * @param localDate
19      * @param country
20      * @param numberOfInfected
21      * @param numberOfDeaths
22      */
23     public CovidLocationStats(LocalDate localDate, String country, int numberOfInfected, int
24         numberOfDeaths) {
25         this.localDate = localDate;
26         this.country = country;
27         this.numberOfInfected = numberOfInfected;
28         this.numberOfDeaths = numberOfDeaths;
29     }
30
31     /**
32      * This function returns the local date.
33      * @return LocalDate
34      */
35     public LocalDate getLocalDate() {
36         return localDate;
37     }
38
39     /**
40      * This function returns the country.
41      * @return country
42      */
43     public String getCountry() {
44         return country;
45     }
46
47     /**
48      * This function gets the number of deaths
49      * @return
50      */
51     public int getNumberOfDeaths() {
52         return numberOfDeaths;
53     }
54
55     /**
56      * This function returns information about the covid location stats.
57      * @return a string of information.
58      */
59     @Override
60     public String toString() {
61         return localDate + ", " + country + ", Infected: " + numberOfInfected + ", Deaths: " +
62             numberOfDeaths;
63     }
64 }
```



(b) Oppgave 3 b)

Er det noen av verdiene til parametrene som inngår i konstruktøren til klassen din som burde sjekkes i forhold til størrelse/tallområde/verdi?  
Har du i så fall noen tanker om hvordan du kan håndtere en slik kontroll (både hvordan sjekke men også hvordan håndtere dersom feil verdi gis)?

Skriv ditt svar her...

LocalDate localDate - burde sjekke om år er fra 2019 eller senere. Ettersom Covid brøt ut i slutten av 2019. Før dette er unødvendig. Måneden må være mellom 1 og 12. Dag må være mellom 1 og 31.

String country, int numberOfInfected og int numberOfDeaths blir også sjekket om de er gyldige via en egen klasse kalt InputHandler.

(c) Oppgave 3 c)

Opprett et register for å holde på COVID-19 registreringene i applikasjonen din. Du velger selv hvilken klasse fra rammeverket med samlinger (*collection framework*) i Java du bruker. **Begrunn valget ditt** med en kommentar i koden.  
Registeret bør som minimum ha følgende funksjonalitet:

- Legge til en COVID-19 registrering
- Søke etter COVID-19 registrering basert *på dato* (returnerer første registrering for gitt dato)
- Søke etter samtlige COVID-19 registreringer *etter en gitt dato* (her skal metoden returnere en samling av registreringer som oppfyller søkekriteriet)
- Regner ut samlet antall døde i et gitt land
- Returnere en *iterator* som andre objekter kan benytte for å iterere over COVID-registreringene i registeret.
- Returnere antallet COVID-19 registreringer i registeret.

Løs oppgaven i ditt IDE (BlueJ, Netbeans, IntelliJ), og kopier deretter koden for hele klassen inn i svarfeltet under.  
**Lim inn koden for registeret ditt her...**

```
1 package me.ntnu.candidate.exam;
2
3 import java.time.LocalDate;
4 import java.time.Month;
5 import java.util.ArrayList;
6 import java.util.HashSet;
7 import java.util.Iterator;
8
9 /**
10  * This class is used to manage and run a Covid Location Stats-registry application.
11  * @Author: 10009
12  */
13 public class CovidRegistry {
14
15     //Using HashSet because a location stat is uniq and a hashset doesn't allow duplicate values.
16     private HashSet<CovidLocationStats> covidRegister;
17
18     /**
19      * This is a constructor.
20      */
21     public CovidRegistry() {
22         covidRegister = new HashSet<>();
23     }
24
25     /**
26      * Helper method for initializing objects of class CovidLocationStats.
27      */
28     public void initializeCovidLocationStats(){
29         covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.JANUARY, 19), "CHINA",
30             136, 1));
31         covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.FEBRUARY, 05), "CHINA",
32             3872, 66));
```

```
31 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 7), "NORGE", 3, 0
32 ));
33 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 9), "USA", 136, 4
34 ));
35 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 9), "CHINA", 136,
36 23));
37 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 22), "NORGE", 136,
38 8));
39 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 24), "USA", 136,
40 119));
41 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.MARCH, 25), "CHINA", 136,
42 4));
43 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.APRIL, 06), "NORGE", 136,
44 3));
45 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.APRIL, 10), "USA", 136,
46 2087));
47 covidRegister.add(new CovidLocationStats(LocalDate.of(2020, Month.APRIL, 10), "CHINA", 136,
48 1));
49 }
50
51 /**
52  * Adds a CovidLocationStats-object to the HashSet covidRegister.
53  * @param covidLocationStatsToBeAdded
54  * @return boolean with info if object was added or not.
55  */
56 public boolean regNewCovidLocationStat(CovidLocationStats covidLocationStatsToBeAdded){
57
58     for(CovidLocationStats list : covidRegister){
59         if(list.equals(covidLocationStatsToBeAdded)){
60             return false;
61         }
62     }
63     covidRegister.add(covidLocationStatsToBeAdded);
64     return true;
65 }
66
67 /**
68  * Searches for the first registration based on input date from user.
69  * @param toBeFound
70  * @return String with stat-info
71  */
72 public String searchFirstRegistrationsByDate(CovidLocationStats toBeFound){
73
74     for(CovidLocationStats list : covidRegister){
75         if(list.getLocalDate().equals(toBeFound.getLocalDate())){
76             return list.toString();
77         }
78     }
79     return null;
80 }
81
82 /**
83  * Searches for all CovidLocationStats after a given date from user.
84  * @param covidRegistrationDate
85  * @return ArrayList with all CovidLocationStats-objects.
86  */
87 public ArrayList <CovidLocationStats> searchAllByAfterDate(LocalDate covidRegistrationDate){
88     ArrayList<CovidLocationStats> obtainedCasesList = new ArrayList<>();
89
90     for(CovidLocationStats obtainedCases : covidRegister){
91         if(obtainedCases.getLocalDate().equals(covidRegistrationDate) || obtainedCases
92             .getLocalDate().isAfter(covidRegistrationDate)){
93             obtainedCasesList.add(obtainedCases);
94         }
95     }
96     return obtainedCasesList;
97 }
98
99 /**
100  * Calculates the total amount of deceased in a given country from user.
101  * @param toBeFound
102  * @return int with number of deceased.
103  */
104 public int numberOfDeceased(CovidLocationStats toBeFound){
105     int totalInSpecificCountry = 0;
106
107     for(CovidLocationStats list : covidRegister){
108         if(list.getCountry().equalsIgnoreCase(toBeFound.getCountry())){
109             totalInSpecificCountry += list.getNumberOfDeaths();
110         }
111     }
112
113     return totalInSpecificCountry;
114 }
115
116 /**
117  * Lists the total amount of entries in the register.
118  * @return int with amount of entries
119  */
120 public int listTotalAmountOfEntries(){
121     return covidRegister.size();
122 }
```

```
113
114  /**
115   * Converts the contents of an ArrayList to a string.
116   * @param listToPrint ArrayList with results from a search.
117   */
118  public void printResult(ArrayList<CovidLocationStats> listToPrint){
119
120      if(listToPrint.isEmpty()){
121          System.out.println("No stats were found");
122      }else{
123          for(CovidLocationStats list : listToPrint){
124              System.out.println(list.toString());
125          }
126      }
127
128  }
129
130  /**
131   * Creates an iterator
132   * @return iterator over HashSet covidRegister
133   */
134  public Iterator<CovidLocationStats> getIterator(){
135      return covidRegister.iterator();
136  }
137
138  /**
139   * Creates a list with all the properties within the registry.
140   * @return ArrayList with all Property-Objects.
141   */
142  public void listAllCovidLocationStats(){
143      Iterator it = getIterator();
144      while(it.hasNext()){
145          System.out.println(it.next().toString());
146      }
147  }
148
149 }
```



4



## Oppgave 4 - Brukergrensesnitt (ca 35%)

I de neste oppgavene skal du ferdigstille applikasjonen ved å lage et enkelt brukergrensesnitt. Du finner eksempelkode du kan ta utgangspunkt i under **vedlegg** helt til slutt i dette oppgavesettet (2 alternativer; et **tekstbasert** og et som bruker **JOptionPane**).  
Lag en egen klasse som representerer brukergrensesnittet til applikasjonen din.  
Her er kravene som skal oppfylles:

### Funksjonalitet

Applikasjonen skal ha følgende funksjonalitet:  
Brukeren må kunne

- registrere/legge inn COVID-19 tilfeller, hver registrering er pr dag.
- skrive ut liste over alle registreringer
- søke etter en registrering basert på dato
- søke etter registreringer *etter* en gitt dato
- regne ut samlet antall døde for et gitt land

#### (a) Oppgave 4 a)

Fullfør koden for klassen med alle funksjonene fra kravspesifikasjonen. Husk god brukervennlighet (tydelige beskjeder til bruker, oversiktlige presentasjoner for bruker osv.)

Når du har gjort det, kopierer du koden til **hele klassen** og limer inn i svarfeltet under.

Lim inn koden fra hele klassen her...

```
1 import me.ntnu.candidate.exam.CovidLocationStats;
2 import me.ntnu.candidate.exam.CovidRegistry;
3 import me.ntnu.candidate.exam.InputHandler;
4
5 import java.time.LocalDate;
6 import java.util.Scanner;
7
8 /**
9  * This class is used to manage a text-based user interface for a Covid Location Stats-registry
10  * application.
11  * @Author: 10009
12  * @version 1.0
13  */
14 public class App {
15
16     private final int ADD_COVID_LOCATION_STAT = 1;
17     private final int SEARCH_ONE_STAT_BASED_ON_DATE = 2;
18     private final int SEARCH_ALL_STATS_BY_AFTER_DATE = 3;
19     private final int CALCULATE_AMOUNT_OF_DEATHS_IN_A_COUNTRY = 4;
```

```
19 private final int PRINT_TOTAL_AMOUNT_OF_COVID_CASES = 5;
20 private final int PRINT_ALL_INFO = 6;
21 private final int EXIT = 7;
22
23 private CovidRegistry covidRegistry;
24 private InputHandler inputHandler;
25 private Scanner scn;
26
27 /**
28  * Constructor that initializes a Scanner and objects of classes.
29  * Also takes the user to the start-menu.
30  */
31 public App() {
32     covidRegistry = new CovidRegistry();
33     inputHandler = new InputHandler();
34     scn = new Scanner(System.in);
35     covidRegistry.initializeCovidLocationStats();
36     start();
37 }
38
39 /**
40  * Creates an object of the App class.
41  * Is the first function that runs when the application starts.
42  * @param args
43  */
44 public static void main(String[] args) {
45     App app = new App();
46 }
47
48 /**
49  * Starts the application. This is the main loop of the application,
50  * presenting the menu, retrieving the selected menu choice from the user,
51  * and executing the selected functionality.
52  */
53 public void start() {
54     boolean finished = false;
55     while (!finished) {
56         int menuChoice = this.showMenu();
57         switch (menuChoice)
58         {
59             case ADD_COVID_LOCATION_STAT:
60                 addCovidLocationStatMenu();
61                 break;
62             case SEARCH_ONE_STAT_BASED_ON_DATE:
63                 searchOneStatBasedOnDateMenu();
64                 break;
65             case SEARCH_ALL_STATS_BY_AFTER_DATE:
66                 searchAllStatByAfterDateMenu();
67                 break;
68             case CALCULATE_AMOUNT_OF_DEATHS_IN_A_COUNTRY:
69                 calculateAmountOfDeathsInACountryMenu();
70                 break;
71             case PRINT_TOTAL_AMOUNT_OF_COVID_CASES:
72                 printTotalAmountOfCovidCasesMenu();
73                 break;
74             case PRINT_ALL_INFO:
75                 printAllInfoMenu();
76                 break;
77             case EXIT:
78                 System.out.println("Thank you for using the app!\n");
79                 finished = true;
80                 break;
81             default:
82                 System.out.println("Unrecognized menu selected..");
83                 break;
84         }
85     }
86 }
87
88 /**
89  * Presents the menu for the user, and awaits input from the user. The menu
90  * choice selected by the user is being returned.
91  *
92  * @return the menu choice by the user as a positive number starting from 1.
93  * If 0 is returned, the user has entered a wrong value
94  */
95 private int showMenu() {
96     int menuChoice = 0;
97     System.out.println("\n***** Covid Location Stats Application v0.1 *****\n");
98     System.out.println("1. Add covid location stat");
99     System.out.println("2. Search for first stat on a specific date");
100    System.out.println("3. Search for all stats after a specific date");
101    System.out.println("4. Calculate amount of deaths in a given country");
102    System.out.println("5. Print total amount of Covid-19 registrations in the register");
103    System.out.println("6. Print all info");
104    System.out.println("7. Quit");
105    System.out.println("\nPlease enter a number between 1 and 7.\n");
106    Scanner sc = new Scanner(System.in);
107    if (sc.hasNextInt()) {
108        menuChoice = sc.nextInt();
109    } else {
110        System.out.println("You must enter a number, not text");
111    }
```

```
111         }
112         return menuChoice;
113     }
114
115     /**
116     * This method lets the user input information to register a new location stat. Done with a
117     * Scanner.
118     * All inputs have to be of a certain data-type.
119     */
120     public void addCovidLocationStatMenu(){
121         System.out.println("-----Please enter date:-----");
122
123         System.out.println("Please enter a year.");
124         int year = inputHandler.getIntInput(2019, 2021);
125
126         System.out.println("Please enter a month.");
127         int month = inputHandler.getIntInput(1, 12);
128
129         System.out.println("Please enter a day.");
130         int day = inputHandler.getIntInput(1, 31);
131
132         LocalDate date = LocalDate.of(year, month, day);
133
134         System.out.println("Please enter the name of the country: ");
135         String countryName = scn.nextLine();
136         countryName = inputHandler.checkString(countryName);
137
138         System.out.println("Please enter the number of infected: ");
139         int infected = inputHandler.getIntInput(0, 99999);
140
141         System.out.println("Please enter the number of deaths: ");
142         int deaths = inputHandler.getIntInput(0, 99999);
143
144         if(covidRegistry.regNewCovidLocationStat(new CovidLocationStats(date, countryName, infected
145             , deaths))){
146             System.out.println("New registry was successfully added");
147         }else{
148             System.out.println("The registry was not added");
149         }
150     }
151
152     /**
153     * Lets the user search for stats in the registry. Done with a Scanner.
154     */
155     public void searchOneStatBasedOnDateMenu(){
156         System.out.println("-----Please enter date:-----");
157         System.out.println("Example: 2020, March, 9");
158
159         System.out.println("Please enter a year.");
160         int year = inputHandler.getIntInput(2019, 2021);
161
162         System.out.println("Please enter a month.");
163         int month = inputHandler.getIntInput(1, 12);
164
165         System.out.println("Please enter a day.");
166         int day = inputHandler.getIntInput(1, 31);
167
168         LocalDate date = LocalDate.of(year, month, day);
169
170         CovidLocationStats covidLocationStats = new CovidLocationStats(date, null, 0, 0);
171
172         if(covidRegistry.searchFirstRegistrationsByDate(covidLocationStats) == null){
173             System.out.println("Didn't find any entries.");
174         }else{
175             System.out.println(covidRegistry.searchFirstRegistrationsByDate(covidLocationStats));
176         }
177     }
178
179     /**
180     * Lets the user search for all stats in the registry. Done with a Scanner.
181     */
182     public void searchAllStatByAfterDateMenu(){
183         System.out.println("-----Please enter date:-----");
184         System.out.println("Example: 2020, 03, 09");
185
186         System.out.println("Please enter a year.");
187         int year = inputHandler.getIntInput(2019, 2021);
188
189         System.out.println("Please enter a month.");
190         int month = inputHandler.getIntInput(1, 12);
191
192         System.out.println("Please enter a day.");
193         int day = inputHandler.getIntInput(1, 31);
194
195         LocalDate date = LocalDate.of(year, month, day);
196
197         covidRegistry.printResult(covidRegistry.searchAllByAfterDate(date));
198     }
199
200     /**
201     * Gives the user information about the amount of deaths in a given country.
202     */
203     public void calculateAmountOfDeathsInACountryMenu(){
```



```
202         System.out.println("Please enter the name of the country: ");
203         String countryName = scn.nextLine();
204         countryName = inputHandler.checkString(countryName);
205         System.out.println(covidRegistry.numberOfDeceased(new CovidLocationStats(null, countryName,
206             0, 0)));
206     }
207
208     /**
209     * Prints a number to the user of all Covid entries within the register.
210     */
211     public void printTotalAmountOfCovidCasesMenu(){
212         System.out.println("-----Registrations in the register-----");
213         System.out.println(covidRegistry.listTotalAmountOfEntries());
214     }
215
216     /**
217     * Prints a list to the user of all Covid entries within the register.
218     */
219     public void printAllInfoMenu(){
220         System.out.println("-----All registered Covid stats-----");
221         covidRegistry.listAllCovidLocationStats();
222     }
223
224 }
225
```

(b) **Oppgave 4 b)**

Med bakgrunn i din endelige løsning av kravspesifikasjonen, hvordan vil du si at designet og implementasjonen din er utført i henhold til design-prinsippene **kobling (eng: coupling)**, og **samstemthet (eng: cohesion)** ?

Gi gjerne konkrete eksempler fra din egen kode.

**Skriv ditt svar her**

**High Cohesion**

- Alle klassene inneholder kun metoder/funksjoner som er relevante til deres klasse. Et eksempel på dette er for eksempel at koden for å utføre operasjonen for å legge alle CovidLocationStats-objekter i en liste, ikke ligger i App-klassen som selv har hovedfokus på å kun inneholde brukergrensesnittet. I stedet ligger koden for denne operasjonen i samme klasse som hvor selve registeret i form av en HashSet ligger.
- I klassen CovidRegistry hender det at de forskjellige metodene kan benytte seg av hverandre. Et eksempel på dette er at metoden searchAllByAfterDate() returnerer en ArrayList med mange Covid-caser. Man bruker så metoden printResult() fra samme klasse til å faktisk skrive ut informasjonen fra denne ArrayListen. Man har gjort det på denne måten fordi metoden printResult() faktisk kan printe ut en hvilken som helst ArrayList av typen <CovidLocationStats>.
- Klassene er ikke store og inneholder ikke urelaterte funksjoner.

**Loose Coupling**

Klassene består hovedsakelig av «loose coupling». Et eksempel på dette er at man må benytte seg av aksessor- og mutator-metodene i CovidLocationStats-klassen for å kunne hente ut informasjon om en covid-registrering. Dette ved bruk av tilgangsmodifikatoren «private» på det forskjellige feltene. «Loose coupling» og «good encapsulation» er det samme.

5    **Opplasting av prosjektfil**

Når du er ferdig med din besvarelse så lager du en **ZIP-fil** (IKKE RAR eller annet format!!) av ***hele prosjektmappen din*** med alle undermapper og IDE-spesifikke filer og laster opp ZIP-filen din.

Du lager ZIP-fil av en mappe på følgende måte:

- I windows: høyreklikk på mappen i filutforskeren din, og velg "Send til->Komprimert (zippet) mappe"
- På MacOSX: høyreklikk mappen i Finder, og velg "Komprimer..."

Sjekk at størrelsen på ZIP-filen viser mer enn noen få bytes. Er du i tvil om du har fått ZIP'et alt? Pakk ut ZIP-filen til annet sted på din datamaskin og gå over innholdet. Ser alt greit ut, kan du levere inn ZIP-filen her.

Din fil ble lastet opp og lagret i besvarelsen din.

Last ned

Fjern

Erstatt

|                       |                              |
|-----------------------|------------------------------|
| Filnavn:              | Eksamen10009-IDATG1001.zip   |
| Filtype:              | application/x-zip-compressed |
| Filstørrelse:         | 99.28 KB                     |
| Opplastingstidspunkt: | 14.12.2020 12:05             |
| Status:               | Lagret                       |