



UNIVERSITETET I BERGEN

KANDIDAT

120

PRØVE

# INF101 0 Objektorientert programmering

Emnekode	INF101
Vurderingsform	Skriftlig eksamen
Starttid	24.05.2022 07:00
Sluttid	24.05.2022 12:00
Sensurfrist	--
PDF opprettet	03.05.2024 10:59

**Informasjon om eksamen**

Oppgave	Tittel	Oppgavetype
i	Egenerklæring/ Declaration, INF101	Informasjon eller ressurser
i	Generell info om digital campus eksamen - INF101, vår 22	Informasjon eller ressurser
i	Innleveringsinformasjon	Informasjon eller ressurser

**Flervalg**

Oppgave	Tittel	Oppgavetype
1	Konstruktør	Flervalg
2	Person.java	Paring

**Forklaring**

Oppgave	Tittel	Oppgavetype
3	Likhet	Langsvar
4	Book	Langsvar

**Koding**

Oppgave	Tittel	Oppgavetype
5	Kortbunke	Langsvar
6	Whac-a-mole	Langsvar
7	Lever kodeoppgaver	Filopplasting
8	Poeng fra semesteroppgavene	Muntlig

## 1 Konstruktør

Når man oppretter et nytt objekt av typen *AIPlayer* hvor mange parametere forventer konstruktøren?

```
public class AIPlayer {  
  
    private String name = "AI Player";  
  
    public void talk() {  
        System.out.println(x: "I am a robot brrrr");  
    }  
  
    @Override  
    public String toString() {  
        return name;  
    }  
}
```

Velg ett alternativ:

☐ 2

☐ 3

☐ 1

☒ 0

---

Maks poeng: 1

## 2 Person.java

```

1 ▼ public class Person {
2
3     private String name;
4     private int yearOfBirth;
5
6 ▼   public Person(String name, int yearOfBirth) {
7       this.name = name;
8       this.yearOfBirth = yearOfBirth;
9 ▲   }
10
11 ▼  public String getName() {
12      return this.name;
13 ▲  }
14
15 ▼  public int getYearOfBirth() {
16      return this.yearOfBirth;
17 ▲  }
18
19 ▲ }

```

Hvilke av følgende konsepter brukes i klassen *Person*?

	Benyttes	Benyttes ikke
Innkapsling	<input checked="" type="radio"/>	<input type="radio"/>
Uforanderlighet (immutability)	<input checked="" type="radio"/>	<input type="radio"/>
Instansmetoder	<input checked="" type="radio"/>	<input type="radio"/>
Feltvariabler	<input checked="" type="radio"/>	<input type="radio"/>
Generiske typer	<input type="radio"/>	<input checked="" type="radio"/>
Komposisjon	<input checked="" type="radio"/>	<input type="radio"/>
Abstraksjon	<input type="radio"/>	<input checked="" type="radio"/>
Polymorfisme	<input type="radio"/>	<input checked="" type="radio"/>

Koden over er også tilgjengelig i git-repositoriet tilhørende eksamen.



### 3 Likhet

```
1 public class Main {  
2  
3     public static void main(String args[]) {  
4         Person x = new Person("Tor", 1999);  
5         Person y = new Person("To" + "r", 1999);  
6         System.out.println(x == y);  
7     }  
8  
9 }
```

Når koden over blir kjørt, printes "false" i konsollen, selv om den som skrev koden hadde forventet å få "true" fordi objektene tross alt er like. Person-klassen er den samme som i forrige spørsmål. Forklar:

- hvorfor svaret blir false, og
- hva som må gjøres for å sammenligne objekter på en god måte, og
- beskriv med ord hvordan man må endre koden for å oppnå dette.

Koden er tilgjengelig også i repositoryet tilhørende eksamen.

**Skriv ca. 3 avsnitt, ikke mer enn 400 ord.**

**Skriv ditt svar her**

why is it false:

it is false because the program is comparing two objects using ==. this compares the two objects pointers in the stack. not the content of the objects in the heap.

what needs to be done:

to compare the two objects you need to use a .equals() method. which should also be implemented for the object (and a hashCode()) as well)

create two methods .equals and .hashCode. the equals method should check for equality in the field variables we care about.

for example it should check for if the year of birth of the two objects are the same using ==. here you can use == because int is a primitive type.

you also need to check if the names of the two objects are the same. here you need to use .equals because String is not a primitive type.

both name and date of birth can be gotten by calling the getName() and getYearOfBirth() method on the two objects

you also need a hashCode method, but this can be made by the IDE.

Ord: 177

Maks poeng: 10

## 4 Book

```
public static void main(String[] args) {
    Book haroldPoter = new Book(Arrays.asList(
        new Page(text: "Why doesn't Voldemort wear glasses?"),
        new Page(text: "Nobody nose.")
    ));

    for (Page page : haroldPoter) {
        String text = page.getText();
        System.out.println(text);
    }
}
```

Kodesnutten over har en kompileringsfeil. Forklar:

- hvorfor denne feilen oppstår, og
- hva man kan gjøre for å fikse den uten å endre noe av koden i main, og
- beskriv med egne ord stegene i løsningen din.

Under ser du koden i Book.java

```
1 ▼ public class Book {
2
3     private List<Page> pages;
4
5 ▼   public Book(List<Page> pages) {
6       this.pages = pages;
7 ▲   }
8
9 ▼   public int length() {
10       return pages.size();
11 ▲   }
12
13 ▼   public Page getPage(int pageNumber) {
14       if (pageNumber >= length())
15           throw new IndexOutOfBoundsException("Index out of bounds");
16
17       return pages.get(pageNumber);
18 ▲   }
19
20 ▲ }
```

Du finner koden i Gitlab-repositoriet i pakken *inf101v22.book*, men for denne oppgaven trenger du **IKKE endre på koden**. Dette er en forklaringsoppgave, og du skal svare her under.

**Repository:** <https://git.app.uib.no/ii/inf101/22v/exam>

**Skriv ca. 3 avsnitt, ikke mer enn 400 ord.**

**Skriv svaret ditt her...**

the compilation error occurs because the Book object is not an object that can be iterated over in a for loop. it does not know by default that it is supposed to give pages to the for loop.

to fix this we need to tell the book that it should give the pages to the loop when it loops over the book.

we can do this by implementing the `Iterable<Page>` interface, and letting the IDE add the iterator method that the interface requires.

the method should return `Iterator<Page>`, which is the iterator for the list of pages in the book, which we can get by calling `.iterator()` on the list of pages.

Ord: 112

---

Maks poeng: 10



## 5 Kortbunke

I en standard (fransk) kortstokk har hvert kort

- en av 4 mulige "*farger*" (suits): kløver, ruter, hjerter eller spar; og
- en av 13 mulige *valører* (ranks): to, tre, fire, fem, seks, sju, åtte, ni, ti, knekt, dronning, konge eller ess.

Det finnes dermed totalt 52 ulike kort. I repositoriet finner du en kort-klasse *Card* som modellerer et slikt kort.

En kortbunke er en samling med kort.

I denne oppgaven skal du implementere en kortbunke på to forskjellig måter. La begge klassene du oppretter være i pakken *inf101v22.cards*.

1. Implementer en kortbunke-klasse *InheritedArrayCardPile* ved bruk av **arv** fra *ArrayList*.
2. Implementer en kortbunke-klasse *ComposedArrayCardPile* ved bruk av **komposisjon** av *ArrayList*.

For begge klassene skal du:

- Implementere en metode *createFullDeck()* som returnerer en ny bunke med kort. Metoden skal returnere et nytt kortbunke-objekt med alle de 52 forskjellige kortene. Burde denne metoden være static eller non-static?
- Implementere en metode med signatur *boolean add(Card)* som legger til et kort i bunken. Metoden skal kaste *IllegalArgumentException* dersom noen forsøker å legge til null, men skal ellers legge til kortet i bunken og alltid returnere true (rart, men slik er spesifikasjonene). Burde denne metoden være static eller non-static?

Man kan se for seg at en klasse som representerer en bunke med kort burde ha flere metoder for å være en fullgod modell, men i denne oppgaven trenger du kun å implementere de to nevnte metodene.

**Klon Gitlab-repositoriet, skriv koden og zip prosjektmappen når du er ferdig med alle kodeoppgavene. Du laster opp zip-filen på siste spørsmål.**

**Repository:** <https://git.app.uib.no/ii/inf101/22v/exam>

**Eventuelle kommentarer på oppgaven kan gis i tekstfeltet nedenfor.**

Ord: 0

---

Maks poeng: 10

## 6 Whac-a-mole

I tivoli-spillet whac-a-mole er poenget å banke ned så mange mulvarper som mulig i løpet av en gitt tid. Hver gang en mulvarp blir banket, dukker det opp en ny mulvarp et tilfeldig sted, som så skal bankes ned. Dersom man banker ned veldig mange mulvarper, kan man vinne en premie.

I denne oppgaven skal du lage en digital versjon av whac-a-mole. Du bør følge instruksjonene så tett som mulig slik at du demonstrerer at du forstår hva som menes.

Instruksjonene og koden finner du i Gitlab-repositoriet.

**Klon Gitlab-repositoriet, skriv koden og zip prosjektmappen når du er ferdig med alle kodeoppgavene. Du laster opp zip-filen på siste spørsmål.**

**Repository:** <https://git.app.uib.no/ii/inf101/22v/exam>

**Eventuelle kommentarer på oppgaven kan gis i tekstfeltet nedenfor.**

Ord: 0

---

Maks poeng: 35

## 7 Lever kodeoppgaver

Last opp prosjektmappen som en zip-fil. Det er mappen som inneholder *pom.xml*.

Viktig: Mappen må være zippet som en zip-fil. Å bruke andre formater, som for eksempel *.tar.gz*, *.7z*, *.rar*, *.war* eller *.jar* vil gjøre at **du mister alle poengene på kodeoppgavene!**



Din fil ble lastet opp og lagret i besvarelsen din.



Last ned



Fjern



Erstatt

Filnavn:	Felix.Kaasa_inf101v22exam.zip
----------	-------------------------------

Filtype:	application/x-zip-compressed
----------	------------------------------


Filstørrelse:	1.18 MB
---------------	---------

Opplastingstidspunkt:	24.05.2022 11:41
-----------------------	------------------

Status:	Lagret
---------	--------

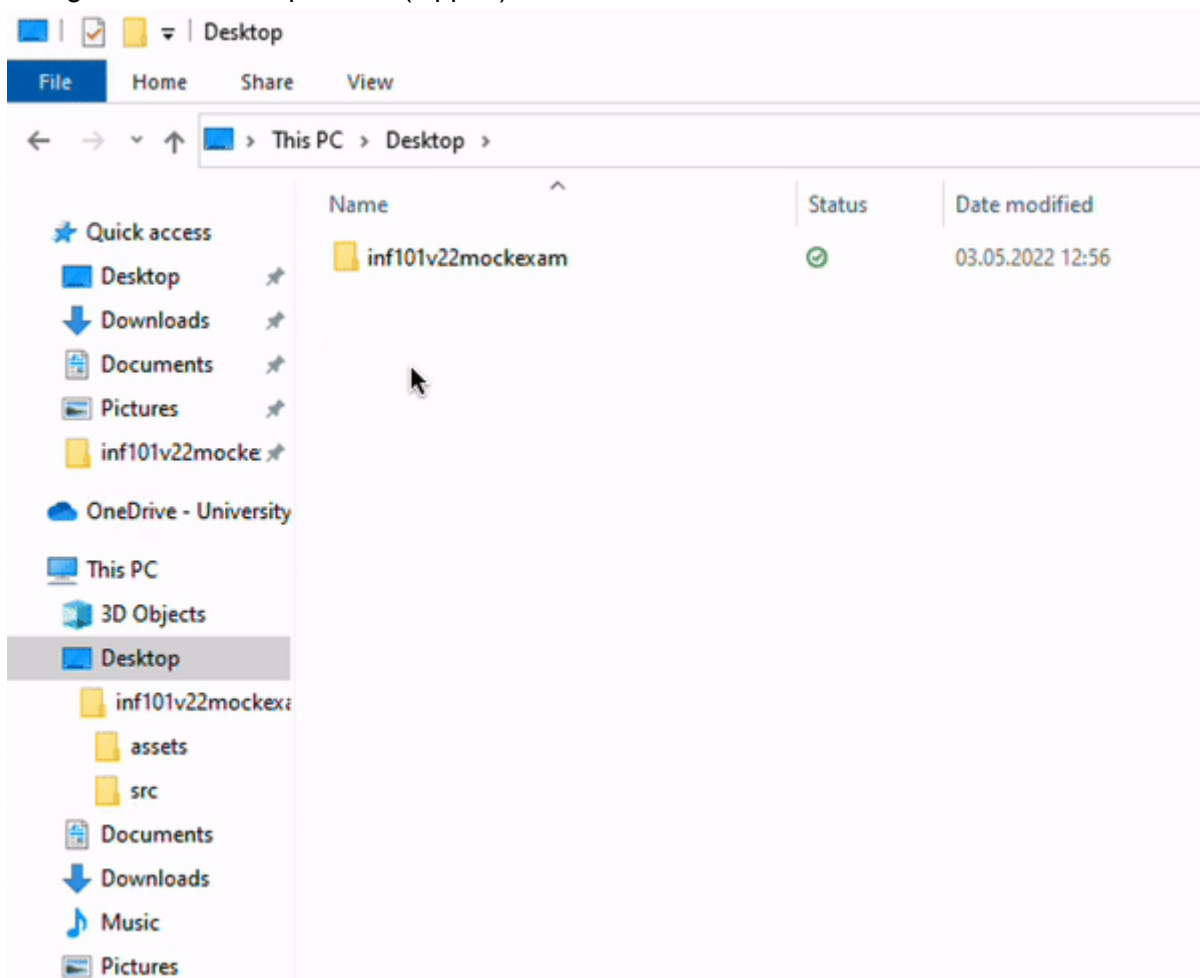
### Hvordan lage .zip på Mac

- Høyreklikk riktig mappe (altså mappen som inneholder pom.xml)
- Velg compress

Name	Date Modified	Size	Kind
>  inf101v22mockexam	Today at 12:31		-- Folder

## Hvordan lage .zip på Windows

- Høyreklikk riktig mappe (altså mappen som inneholder pom.xml)
- Velg send to -> compressed (zipped) folder



## Hvordan lage .zip på Ubuntu

- Høyreklikk riktig mappe (altså mappen som inneholder pom.xml)
- Velg compress
- I menyen som dukker opp, velg .zip

---

Maks poeng: 0

## **8 Poeng fra semesteroppgavene**

Dine poeng fra semesteroppgavene vil legges til her. Det er ingen ting du kan gjøre fra eller til på dette punktet nå.

---

Maks poeng: 30