Høgskolen i Østfold

# EKSAMEN

| Emnekode:<br>ITF10611 | Emne:<br>Objektorientert Programmering |
|---|---|
| Dato: 26 mai 2015 | Eksamenstid: kl. 09:00 til kl. 13:00 |
| Hjelpemidler:<br>To A4-ark (fire sider) med egne notater | Faglærer:<br>Per Bisseberg |

Eksamensoppgaven:
Oppgavesettet består av 17 sider inklusiv denne forsiden og vedlegg. Kontroller at oppgaven er komplett før du begynner å besvare spørsmålene. Du er selv ansvarlig for å kontrollere at oppgavesettet er komplett. Les gjennom alle oppgavene før du begynner. Det er på hver hovedoppgave angitt hvor mye disse teller av totalen. Karakter fastsettes på grunnlag av en helhetsvurdering av besvarelsen.

I oppgavene hvor du blir bedt om å skrive kode anbefales det at du skriver løsningen med java-syntaks. Er du derimot usikker på hvordan du skal besvare en oppgave med kode kan du skrive svaret med egne ord, det er da viktig at du beskriver logikken på en omfattende og detaljert måte.

Pass også på å svare på alle oppgaver. Det er bedre å skrive litt i grove trekk hvordan du ser for deg at oppgaven kan løses, enn å ikke skrive noe i det hele tatt, dersom du står fast.

Det ligger utdrag fra Java 8 Dokumentasjonen sist i vedlegget.

Ønsker dere alle en riktig god sommer, og takk for et virkelig hyggelig semester :-)

***Lykke til!***

Sensurdato: 17. juni 2015
Karakterene er tilgjengelige for studenter på studentweb senest 2 virkedager etter oppgitt sensurfrist. Følg instruksjoner gitt på: www.hiof.no/studentweb

## Oppgave 1 (25%)

Disse oppgavene skal besvares kort og presis. Du trenger altså ikke skrive en liten stil på hver av dem, men pass på at du besvarer alt oppgaven spør etter. Det vil bli lagt vekt på at forklaringen er skrevet med dine ord, og at den ikke er avskrift fra andre kilder. Det er fordelaktig å lage eksempler ved kode og/eller illustrasjoner.

### Oppgave 1.1

a) Forklar begrepet abstraksjon.
b) Lag en illustrasjon som viser abstraksjon.
c) Lag enkel kode som viser abstraksjon.

### Oppgave 1.2

Forklar disse nøkkelordene fra Java

a) new
b) static
c) this

### Oppgave 1.3

Forklar og eksemplifiser begrepet innkapsling.

## Oppgave 2(25%)

Analyse av kode.

### Oppgave 2.1(10%)

Disse oppgavene inneholder en eller flere feil som gjør at koden ikke kompilerer. Du skal skrive hva kompileringsfeilen(e) er og skrive koden som skal til for at koden skal kompilere.

#### Oppgave 2.1.1

```
public class A {
	private static final String NAVN = "superduperklasse";

	public A(String navn) {
		this.NAVN = navn;
	}
}

public class B extends A{

	private int verdi;

	public B(String navn, int verdi) {
		super(navn);
		this.verdi = verdi;
	}
}
```

Oppgave 2.1.2

```
public abstract class  A {
	private String tekst;

	public A(String tekst) {
		this.tekst = tekst;
	}

	public abstract void printTekst();
}


public class B implements A{
	private String navn;

	public B(String tekst, String navn) {
		super(tekst);
		this.navn = navn;
	}
}
```

## Oppgave 2.2(15%)

I disse deloppgavene skal du skrive hvilken utskrift koden i mainmetoden resulterer i.

### Oppgave 2.2.1

```
public class Person {
	private String fornavn;
	private String etternavn;
	private int alder;

	public Person(String fornavn, String etternavn, int alder) {
		this.fornavn = fornavn;
		this.etternavn = etternavn;
		this.alder = alder;
	}

	@Override
	public String toString() {
		return fornavn + " " + etternavn + ", " + alder;
	}

	// get og set ikke tatt med, men finnes for alle felt.

}


// main-metode
public static void main(String[] args) {
	ArrayList<Person> personer = new ArrayList<Person>();

	Person anne = new Person("Anne", "Olsen", 32);
	personer.add(anne);
	personer.add(anne);

	personer.get(1).setAlder(42);

	for(Person p: personer){
		System.out.println(p);
	}
}
```

Oppgave 2.2.2

Vi benytter Person-klassen fra forrige oppgave

```
public static void main(String[] args) {
	ArrayList<Person> personer = new ArrayList<Person>();
	personer.add(new Person("Anne", "Olsen", 32));
	personer.add(new Person("Ole", "Pettersen", 34));
	personer.add(new Person("Nina", "Hansen", 29));

	Collections.sort(personer,new Comparator<Person>() {

		@Override
		public int compare(Person p1, Person p2) {
			return p2.getEtternavn().length() - p1.getEtternavn().length();
		}
	});

	for(Person p: personer){
		System.out.println(p);
	}
}
```

Oppgave 2.2.3

```
public class Linje {
	private String tekst;

	public Linje(String tekst) {
		this.tekst = tekst;
	}

	@Override
	public String toString() { return tekst; }

	public String getTekst() { return tekst; }

	public void setTekst(String tekst) { this.tekst = tekst; }
}

public class Fildata {
	public static ArrayList<Linje> fillinjer = new ArrayList<Linje>();

	public static ArrayList<Linje> genererParagrafer(String tegn){
		ArrayList<Linje> ret = new ArrayList<Linje>();
		for(Linje l: fillinjer){
			String[] mid = l.getTekst().split(tegn);
			String nyTekst = "<p>";
			for(int i = 0; i < mid.length; i++){
				if(i+1 != mid.length){
					nyTekst += mid[i] + " ";
				}
				else{
					nyTekst += mid[i] + "</p>";
				}
			}
			ret.add(new Linje(nyTekst));
		}
		return ret;
	}
}

// main-metode
public static void main(String[] args) {
	Fildata.fillinjer.add(new Linje("342523;Per;Persen;3.7;180"));
	Fildata.fillinjer.add(new Linje("231252;Ida;Idasen;1.4;35"));
	Fildata.fillinjer.add(new Linje("932532;Tore;Torsen;4.1;180"));

	for(Linje l: Fildata.genererParagrafer(";")){
		System.out.println(l);
	}
}
```

## Oppgave 3(50%)

Vi skal utvikle en prototype for en oppdragsgiver, Høgskolen i Østfold. Vi har nettopp begynt utviklingen og vi jobber med å få på plass kjernefunksjonalitet. Vedlagt finner du de klasser og interface vi har så langt.

HiØ trenger et nytt romreservasjonssystem. Systemet skal håndtere alle personer som kan kunne trenge å benytte et slikt system, altså både studenter og ansatte. I tillegg så må systemet støtte opprettelsen av rom, men høgskolen har flere typer rom og forskjellig regelverk knyttet til hvem som kan reservere rommene. Romtypene vi skal ta med i prototypen er følgende: forelesningsrom, møterom og grupperom, det er høyst sannsynlig at programmet må kunne støtte andre typer av rom senere. Reglene rundt reservasjon er som følger:

- Grupperom
    - Kan reserveres av studenter og ansatte.
    - Studenter kan maksimalt reservere et spesifikt grupperom 2 timer per reservasjon.
    - Ansatte har ingen tidsbegrensing.
- Møterom
    - Kan kun reserveres av ansatte.
    - Kan kun reservere et gitt møterom inntil fire timer per reservasjon.
- Forelesningsrom
    - Kan kun reserveres av ansatte.
    - En ansatt skal kunne reservere et forelesningsrom flere for dager i en operasjon.
        - Dvs. at en ansatt skal kunne reservere et forelesningsrom for mange dager av gangen.

### Oppgave 3.1

Lag et klassediagram, det er ikke nødvendig å ta med felter og metoder, over alle klassene og interface (se vedlegg) i systemet hvor du tydelig får frem eventuelle arverelasjoner, implementasjoner og assosiasjoner. Assosiasjonene skal vises med aggregering-/komposisjons-«piler». Du trenger ikke tegne inn klassen Main.

### Oppgave 3.2

a) Skriv ferdig konstruktørene i klassene Ansatt og Student.

b) Den abstrakte klassen Rom definerer den abstrakte metoden ***reservasjon(Person person, Tidsrom tidsrom).*** Implementer denne metoden der det er påkrevd og skriv koden som utfører reservasjon etter reglene som er beskrevet over.

c) Vi har interfacet «Multireserverbar», implementer denne i klassen Forelesningsrom.

- Du skal her skrive koden som gjør at en ansatt kan reservere et rom mange ganger.

## Oppgave 3.3

I klassen Rom har vi metoden printReservasjoner()
Denne metoden skal skrive ut alle reservasjoner i Reservasjonsoversikten for ett rom. Gjør nødvendige tillegg slik at utskriftene fra denne metoden ser slik ut:

```
Reservert av: Per Bisseberg. Fra: Thu Jan 15 10:15:00 CET 2015 Til:
Thu Jan 15 11:00:00 CET 2015

Reservert av: Tom Heine Nätt. Fra: Fri Jan 16 10:15:00 CET 2015 Til:
Fri Jan 16 11:00:00 CET 2015
```

## Oppgave 3.4

I klassen Main finner vi en metode som lar en ansatt gjenta en reservasjon ukentlig. Her finner du en rekke kommentarer som dette: // A Du skal erstatte disse kommentarene (A - G) i metoden med egne kommentarer som forklarer og beskriver de underliggende operasjonene.

## Oppgave 3.5

I klassen Main ar vi en funksjonen printAllRomInfo() som inneholder nødvendig funksjonalitet for å skrive ut all informasjon om alle rom vi har i systemet. Vi ønsker derimot å kunne sortere rommene på følgende måte før vi skriver ut rominformasjonen:

a) Gjør tillegg i klassen Rom som definerer at den naturlige sorterings-ordenen på romobjekt er på synkende antall plasser.
b) Skriv koden i main-metoden som utfører sortering og utskrift av all rom informasjon.

# Vedlegg til oppgave 3

Klasser:

- Rom
    - Indre klasse: Reservasjon
- Forelesningsrom
- Møterom
- Grupperom
- Person
- Ansatt
- Student
- Tidsrom
- Main – Trenger ikke være med i oppgave 3.1 UML

Interface:

- multireserverbar

Vedlegget er på 8 sider inklusive denne siden.

## Rom

```
public abstract class Rom {
	private String navn;
	private int antallPlasser;
	private static ArrayList<Rom> rom = new ArrayList<Rom>();
	private ArrayList<Reservasjon> romreservasjoner =
		new ArrayList<Reservasjon>();

	// konstruktør
	public Rom(String navn, int antallPlasser){
		this.navn = navn;
		this.antallPlasser = antallPlasser;
		rom.add(this);
	}

	// abstrakt metode for utførelse av reservasjon
	abstract public void reservasjon(Person person, Tidsrom tidsrom);

	// tekstlig representasjon av Rom-objektet
	@Override
	public String toString() {
		return "Rom" + navn + "\tPlasser: " + antallPlasser;
	}

	// Du kan forutsette at alle private felt har get og set tilgjengelig


	//**************** Reservasjoner ****************
	// metode for opprettelse av ny reservasjon for dette rommet
	protected void leggTilReservasjon(Person person, Tidsrom tidsrom){
		// sjekker om rommet allerede er reservert
		if(!erReservert(tidsrom)){
			romreservasjoner.add(new Reservasjon(this, tidsrom, person));
		}
	}

	// sjekker om rommet er reservert i dette tidsrommet
	private boolean erReservert(Tidsrom tidsrom){
		for(Reservasjon r: romreservasjoner){
			if(r.tidsrom.konflikt(tidsrom)){
				System.out.println("Rommet er allered reservert");
				return true;
			}
		}
		return false;
	}

	// skriver ut alle reservasjoner for dette rommet
	public void printReservasjoner(){
		for(Reservasjon r: romreservasjoner){
			System.out.println(r);
		}
	}
```

```
	// indre klasse: Reservasjon
	private class Reservasjon {
		private Rom rom;
		private Tidsrom tidsrom;
		private Person reservator;

		// konstruktør
		public Reservasjon(Rom rom, Tidsrom tidsrom,
				Person reservator) {
			this.rom = rom;
			this.tidsrom = tidsrom;
			this.reservator = reservator;
		}

		// tekstlig representasjon av Reservasjon-objektet
		@Override
		public String toString() {
			return "Reservert av: " + reservator + " " + tidsrom;
		}

		// get- og setmetoder er utelatt da de ikke er nødvendige
	}
}
```

### Forelesningsrom

```
public class Forelesningsrom extends Rom{

	// konstruktør
	public Forelesningsrom(String navn, int antallPlasser){
		super(navn, antallPlasser);
	}
}
```

### Møterom

```
public class Møterom extends Rom {

	// konstruktør
	public Møterom(String navn, int antallPlasser) {
		super(navn, antallPlasser);
	}
}
```

### Grupperom

```
public class Grupperom extends Rom {

	// konstruktør
	public Grupperom(String navn, int antallPlasser) {
		super(navn, antallPlasser);
	}
}
```

### Person

```
public abstract class Person {
	private String fornavn;
	private String etternavn;
	private String avdeling;
	private static ArrayList<Person> personer = new ArrayList<Person>();

	// konstruktør
	public Person(String fornavn, String etternavn, String avdeling) {
		this.fornavn = fornavn;
		this.etternavn = etternavn;
		this.avdeling = avdeling;
		personer.add(this);
	}

	// Du kan forutsette at alle private felt har get og set tilgjengelig
}
```

### Ansatt

```
public class Ansatt extends Person {
	private String ansattId;

	// konstruktør
	// oppgave 3.2a løses her


	// Du kan forutsette at alle private felt har get og set tilgjengelig
}
```

### Student

```
public class Student extends Person {
	private int studentnummer;

	// konstruktør
	// oppgave 3.2a løses her


	// Du kan forutsette at alle private felt har get og set tilgjengelig
}
```

## Tidsrom

```
public class Tidsrom {
	private Calendar fra;
	private Calendar til;

	//konstruktør
	public Tidsrom(Calendar fra, Calendar til) {
		// Sjekker om fra tiden er før til tiden
		if(fra.before(til)){
			this.fra = fra;
			this.til = til;
		}
		else{
			// feilmelding og lar feltene være null (ikke godt håndtert..
			men dette er bare en prototype)
			System.out.println("Til dato kan ikke være eldre enn fra dato");
		}
	}

	// sjekker om det er tidskonflikt med tidsrommet og dette rommet
	public boolean konflikt(Tidsrom t){
		if(t.fra.after(til)){
			return false;
		}
		return true;
	}

	// tekstlig representasjon av objektet
	@Override
	public String toString() {
		return "Fra: " + fra.getTime() + " - Til: " + til.getTime();
	}

	// henter varighet i timer for dette tidsrommet
	public int hentVarighet(){
		Duration d = Duration.between(fra.toInstant(), til.toInstant());
		return (int) d.toHours();
	}

	// Du kan forutsette at alle private felt har get og set tilgjengelig
}
```

## Main

```
public class Main {

	public static void main(String[] args) {
		// start
		// ikke nødvendig å skrive kode her
	}

	// metode for å lage reservasjoner som gjentar seg i n uker
	public static void ukesReservasjon(Multireserverbar rom, Ansatt ansatt,
			Tidsrom tidsrom, int n){
		// A
		ArrayList<Tidsrom> liste = new ArrayList<Tidsrom>();
		// B
		liste.add(tidsrom);
		// C
		for(int i = 0; i < n; i++){
			// D
			GregorianCalendar fra = new GregorianCalendar(
				tidsrom.getFra().get(Calendar.YEAR),
				tidsrom.getFra().get(Calendar.MONTH),
				tidsrom.getFra().get(Calendar.DAY_OF_MONTH),
				tidsrom.getFra().get(Calendar.HOUR_OF_DAY),
				tidsrom.getFra().get(Calendar.MINUTE)
			);

			GregorianCalendar til = new GregorianCalendar(
				tidsrom.getTil().get(Calendar.YEAR),
				tidsrom.getTil().get(Calendar.MONTH),
				tidsrom.getTil().get(Calendar.DAY_OF_MONTH),
				tidsrom.getTil().get(Calendar.HOUR_OF_DAY),
				tidsrom.getTil().get(Calendar.MINUTE)
			);

			// E
			fra.add(Calendar.WEEK_OF_YEAR, i+1);
			til.add(Calendar.WEEK_OF_YEAR, i+1);

			// F
			liste.add(new Tidsrom(fra, til));
		}
		// G
		rom.multireservasjon(ansatt, liste);
	}

	// metode for å skrive ut info om alle rom
	public static void printAllRomInfo(){
		for(Rom r: Rom.getRom()){
			System.out.println(r);
		}
	}
}
```

## Multireserverbar

```
public interface Multireserverbar {
	// metode for ukentlige reservasjoner
	void multireservasjon(Ansatt ansatt, ArrayList<Tidsrom> liste);
}
```

## Utdrag fra Java 8 Dokumentasjonen

### Calendar

*Felt/Field:*

`WEEK_OF_YEAR`

Field number for `get` and `set` indicating the week number within the current year.

*Metoder:*

`add(int field, int amount)`

Adds or subtracts the specified amount of time to the given calendar field, based on the calendar's rules.

`get(int field)`

Returns the value of the given calendar field.

### GregorianCalendar

*Konstruktør:*

`GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute)`

Constructs a `GregorianCalendar` with the given date and time set for the default time zone with the default locale.