

**EKSAMEN**

|  |  |  |
| --- | --- | --- |
| Emnekode:  ITF10611 | Emne:  Objektorientert programmering | |
| Dato: 12. mai 2014 | Eksamenstid: kl. 09:00 til kl. 13:00 | |
| Hjelpemidler:  To A4-ark (fire sider) med egne notater | | Faglærer:  Per Bisseberg |
| Eksamensoppgaven:  Oppgavesettet består av 24 sider inklusiv denne forsiden og vedlegg. Kontroller at oppgaven er komplett før du begynner å besvare spørsmålene.  Det er på hver hovedoppgave angitt hvor mye disse teller av totalen. Karakter fastsettes dog på grunnlag av en helhetsvurdering av besvarelsen.  Du er selv ansvarlig for å kontrollere at oppgavesettet er komplett. Les gjennom alle oppgavene før du begynner.  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.  Ønsker dere alle en riktig god sommer, og takk for et fabelaktig hyggelig semester :-)  ***Lykke til!*** | | |
| Sensurdato: 3. juni 2014  Karakterene er tilgjengelige for studenter på studentweb senest 2 virkedager etter oppgitt sensurfrist. Følg instruksjoner gitt på: [www.hiof.no/studentweb](http://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**

1. Forklar hva arv er innen objektorientert programmering.
2. Hva oppnår vi ved å benytte arv?
3. Lag en illustrasjon som viser arv.
4. Lag enkel kode som viser hvordan vi implementerer arv i Java.

## **Oppgave 1.2**

Forklar disse nøkkelordene fra Java

1. instanceof
2. final
3. try, catch og finally

## **Oppgave 1.3**

Forklar og eksemplifiser begrepet polymorfisme.

# **Oppgave 2(25 %)**

Her skal du analysere kode. Noen av eksemplene vil inneholde feil, i disse tilfellene skal du skrive hva feilen er og hva som må til for å løse feilen. Finnes det ikke feil i koden skal du skrive hvilken utskrift koden resulterer i.

## **Oppgave 2.1**

**public** **class** A {

**private** String navn;

**public** A(String navn) {

**this**.navn = navn;

}

}

**public** **class** B **extends** A{

**private** **int** verdi;

**public** B(**int** verdi) {

**this**.verdi = verdi;

}

}

## **Oppgave 2.2**

**public** **abstract** **class** A {

**private** **int** verdi;

**public** A(**int** verdi) {

**this**.verdi = verdi;

}

**void** PrintVerdi();

}

## **Oppgave 2.3**

**private class** Person{

**private** String fornavn;

**private** String etternavn;

**public** Person(String fornavn, String etternavn) {

**this**.fornavn = fornavn;

**this**.etternavn = etternavn;

}

**public** String getFornavn() {

**return** fornavn;

}

**public** String getEtternavn() {

**return** etternavn;

}

**public** **void** setFornavn(String fornavn) {

**this**.fornavn = fornavn;

}

**public** **void** setEtternavn(String etternavn) {

**this**.etternavn = etternavn;

}

}

**public** **static** **void** main(String[] args) {

Person p1 = **new** Person("Per", "Bisseberg");

Person p2 = p1;

p2.setEtternavn("Nisseberg");

System.*out*.println(p1.getEtternavn());

System.*out*.println(p2.getEtternavn());

}

## **Oppgave 2.4**

**public** **abstract** **class** Person {

**private** String fornavn;

**private** String etternavn;

**public** Person(String fornavn, String etternavn) {

**this**.fornavn = fornavn;

**this**.etternavn = etternavn;

}

**public** String getFornavn() {

**return** fornavn;

}

**public** String getEtternavn() {

**return** etternavn;

}

}

**public** **class** Ansatt **extends** Person{

**private** **int** ansattID;

**public** Ansatt(String fornavn, String etternavn, **int** ansattID) {

**super**(fornavn, etternavn);

**this**.ansattID = ansattID;

}

@Override

**public** String toString(){

**return** ansattID + ": " + getEtternavn() + " " + getFornavn();

}

}

**public** **class** Student **extends** Person{

**private** String studID;

**public** Student(String fornavn, String etternavn, String studID) {

**super**(fornavn, etternavn);

**this**.studID = studID;

}

@Override

**public** String toString(){

**return** studID + ": " + getFornavn() + " " + getEtternavn();

}

}

Fortsetter på neste side.

**public** **static** **void** main(String[] args) {

LinkedList<Person> personliste = **new** LinkedList<>();

personliste.add(**new** Ansatt("Per", "Biffeberg", 1337));

personliste.add(**new** Student("Kari", "Olsen", "S12314"));

personliste.add(**new** Ansatt("Tom Dag", "Ognatt", 123));

**for**(Person p : personliste){

System.*out*.println(p);

}

}

## **Oppgave 2.5**

**class** Motorsykkel **implements** Comparable<Motorsykkel>{

**private** String navn;

**private** Integer aarsmodell;

**public** Motorsykkel(String navn, **int** aarsmodell) {

**this**.navn = navn;

**this**.aarsmodell = aarsmodell;

}

@Override

**public** **int** compareTo(Motorsykkel o) {

**return** navn.compareTo(o.navn);

}

@Override

**public** String toString() {

**return** navn + " " + aarsmodell;

}

**public** Integer getAarsmodell() {

**return** aarsmodell;

}}

Fortsetter på neste side.

**public** **class** Main {

**static** ArrayList<Motorsykkel> *mcListe* = **new** ArrayList<>();

**public** **static** **void** main(String[] args){

*mcListe*.add(**new** Motorsykkel("Suzuki 1300XF", 2013));

*mcListe*.add(**new** Motorsykkel("Yamaha MT-09", 2014));

*mcListe*.add(**new** Motorsykkel("BMW 1100T", 2011));

Collections.*sort*(*mcListe*);

// skriv ut liste

*printMCliste*();

System.*out*.println();

Collections.*sort*(*mcListe*, **new** Comparator<Motorsykkel>() {

@Override

**public** **int** compare(Motorsykkel mc1, Motorsykkel mc2) {

**return** - mc1.getAarsmodell().compareTo(mc2.getAarsmodell());

}

});

// skriv ut liste

*printMCliste*();

}

**public** **static** **void** printMCliste(){

**for**(Motorsykkel mc : *mcListe*){

System.*out*.println(mc);

}

}

}

### **Oppgave 3(50 %)**

Vi utvikler et system for en oppdragsgiver, BILLIGBIL A/S. 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.

BILLIGBIL A/S selger både nye og brukte biler. De har en stor park med bruktbiler og for å få ekstra inntekter har de begynt å leie ut bruktbilene. De ønsker ett system som holder oversikt over varelageret som består av nye og brukte biler. Vi må kun håndtere salg av nye og brukte biler og utleie av brukte biler.

BILLIGBIL A/S ønsker også å holde på en oversikt over tidligere kunder slik at man kan sende målrettet reklame til disse.

Til slutt ønsker de at systemet tar vare på alle salg i en transaksjonshistorikk. En transaksjon kan være salg eller utleie av en bil til en kunde. Det ønskes at vi kan skille mellom transaksjoner av type utleie og salg. En transaksjon består av en kunde og en gitt bil, om det er et salg inneholder transaksjonen også en salgssum. Salgsummen i en transaksjon er alltid inkludert registreringsavgiften. Avgiften varierer på bakgrunn av om det er en tidligere uregistrert ny bil eller alderen på bruktbilen. I klassen Salg finner du de ulike avgiftene og måter å finne disse.  
  
Når vi selger en bil ønsker vi at denne skal fjernes fra varelageret slik at denne er korrekt på bakgrunn av de salgstransaksjonene som er gjennomført. Vi bryr oss ikke om å ha en oversikt over hvilke biler som er utleid til enhver tid, dette er noe vi skal implementerer i neste omgang.

### **Oppgave 3.1**

Lag et UML-diagram over alle klassene(se vedlegg) i systemet hvor du tydelig får frem eventuelle arverelasjoner. Du trenger ikke tegne inn klassen Main.

### **Oppgave 3.2**

1. Skriv ferdig konstruktørene i klassene Nybil og Bruktbil
2. Vi har interfacet «I\_utleie» implementer dette i klassen Bruktbil.

### **Oppgave 3.3**

Som nevnt over så er en salgssum inkludert nyregistrering eller omregistrering. I dette systemet har en nyregistrering en fast pris på kr 13.798,00 mens en omregistrering har ulik pris basert på alder, se tabell under. I klassen Salg finner du igjen disse.

|  |  |
| --- | --- |
| Alder på bil | Avgift i Kr |
| 2 | 10.329,00 |
| 3 | 7.550,00 |
| 4 – 11 | 4.977,00 |
| 12 + | 1.535,00 |

Den abstrakte klassen Bil definerer den abstrakte metoden ***selg(Kunde kunde, double pris).***

1. Forklar kort hva abstrakte klasser og metoder er.
2. Klassene Nybil og Bruktbil vil ikke kompilere pga feil (selv etter att 0ppgave 3.2 a er løst). Fullfør koden i disse klassene og skriv logikken som skal utføres når vi selger nye og brukte biler.

## **Oppgave 3.4**

I klassen Salg finner du en rekke kommentarer (A - F) bestående av kun en bokstav for eksempel slik:   
// Kommentar F

Du skal bytte ut disse med dine egne kommentarer. Pass på å kommenter den overordnede logikken.

## **Oppgave 3.5**

Vi har en klasse Main som bruker vi til å teste koden vår så langt. I denne klassen skal du gjøre ferdig følgende metoder:

1. skrivUtTransaksjoner()  
   Denne metoden skal skrive ut alle transaksjoner i transaksjonshistorikken. Gjør nødvendige tillegg til klassene Salg og Utleie slik at utskriftene fra denne metoden ser slik ut:  
     
   Utleie av BMW 320i til Per Bisseberg total, utleiepris:2220.0  
   Salg av bruktbil BMW 320i til Per Bisseberg, salgssum:201535.0  
   Salg av nybil Audi A4 til Per Bisseberg, salgssum:303798.0
2. skrivUtVarelager()  
   Denne metoden skal skrive ut informasjon om alle, brukte og nye, biler på lager. Det skal være ulik utskrift om bilen er ny eller brukt. Utskriften skal se slik ut:  
     
   Ny: Audi A4 km:56 årsmodell:2014 pris:298456  
   Brukt: BMW 320i km:56789 årsmodell:2000 pris:234000
3. Vi ønsker å kunne sortere utskriften på varelageret etter synkende pris(høy til lav). Implementer funksjonalitet som lar oss gjøre dette.

# **Vedlegg til oppgave 3**

Klasser:

* Bil
* Nybil
* Bruktbil
* Merke
* Modell
* Kunde
* Transaksjon
* Salg
* Utleie
* Main – Trenger ikke være med i oppgave 3.1 UML

Interface:

* I\_utleie

Vedlegget er på 15 sider inklusive denne siden.

**Bil.java**

/\*\*

\* En bil som kan selges eller leies ut

\* brukt eller ny

\* **@author** Per

\*

\*/

**public** **abstract** **class** Bil {

// varelageret -> alle bilene

**private** **static** ArrayList<Bil> *varelager* = **new** ArrayList<>();

**private** Modell modell;

**private** **int** aarsmodell;

**private** **int** kmStand;

**private** **int** pris;

/\*\*

\* constructor

\* **@param** modell - bilmodellen

\* **@param** aarsmodell - produksjonsår

\* **@param** kmStand - kilometerstand

\* **@param** pris - vinduspris

\*/

**public** Bil(Modell modell, **int** aarsmodell, **int** kmStand, **int** pris) {

**this**.modell = modell;

**this**.aarsmodell = aarsmodell;

**this**.kmStand = kmStand;

**this**.pris = pris;

// legger til bilen i varelageret

*varelager*.add(**this**);

}

// abstrakt selg metode for salg av biler

**public** **abstract** **void** selg(Kunde kunde, **double** pris);

// get og set

**public** Modell getModell() { **return** modell; }

**public** **void** setModell(Modell modell) {

**this**.modell = modell;

}

**public** **int** getAarsmodell() { **return** aarsmodell; }

**public** **void** setAarsmodell(**int** aarsmodell) {

**this**.aarsmodell = aarsmodell;

}

Fortsetter på neste side

**public** **int** getKmStand() { **return** kmStand; }

**public** **void** setKmStand(**int** kmStand) {

**this**.kmStand = kmStand;

}

**public** **int** getPris() { **return** pris; }

**public** **void** setPris(**int** pris) {

**this**.pris = pris;

}

**public** **static** ArrayList<Bil> getLagerliste() {

**return** *varelager*;

}

}

**Nybil.java**

/\*\*

\* en ny bil som kan selges

\* **@author** Per

\*

\*/

**public** **class** Nybil **extends** Bil{

/\*\*

\* constructor

\* **@param** modell - bilmodellen

\* **@param** aarsmodell - produksjonsår

\* **@param** kmStand - kilometerstand

\* **@param** pris - vinduspris

\*/

**public** Nybil(Modell modell, **int** aarsmodell, **int** kmStand, **int** pris)

{

}

}

**Bruktbil.java**

/\*\*

\* en brukbil som kan leies ut eller selges

\* **@author** Per

\*

\*/

**public** **class** Bruktbil **extends** Bil{

**private** **int** antallTideligereEiere;

// antall ganger bruktbilen har vært utleid

**private** **int** antallGangerUtleid;

/\*\*

\* constructor

\* **@param** modell - bilmodellen

\* **@param** aarsmodell - produksjonsår

\* **@param** kmStand - kilometerstand

\* **@param** pris - vinduspris

\* **@param** antallTideligereEiere - hvor mange tideligere eiere

\*/

**public** Bruktbil(Modell modell, **int** aarsmodell, **int** kmStand,

**int** pris, **int** antallTideligereEiere) {

}

// get og set

**public** **int** getAntallTideligereEiere() {

**return** antallTideligereEiere;

}

**public** **void** setAntallTideligereEiere(**int** antallTideligereEiere) {

**this**.antallTideligereEiere = antallTideligereEiere;

}

**public** **int** getAntallGangerUtleid() {

**return** antallGangerUtleid;

}

}

**I\_utleie.java**

/\*\*

\* Interface for utleie

\* **@author** Per

\*

\*/

**public** **interface** I\_utleie {

/\*\*

\* lei ut denne bilen -> utleietransaksjon

\* **@param** kunde - kunden som leier bilen

\* **@param** døgnpris - prisen for å leie bilen i ett døgn

\* **@param** antallDager - antall dager bilen er leid ut

\*/

**void** leiut(Kunde kunde, **double** døgnpris, **int** antallDager);

}

**Merke.java**

/\*\*

\* Et bilmerke feks Volvo

\* **@author** Per

\*

\*/

**public** **class** Merke {

// alle bilmerker i systemet

**private** **static** ArrayList<Merke> *merkeoversikt* =

**new** ArrayList<>();

**private** String navn;

/\*\*

\* constructor

\* **@param** navn - navnet på bilmerket

\*/

**public** Merke(String navn) {

**this**.navn = navn;

// legg til merket i oversikten

*merkeoversikt*.add(**this**);

}

// get og set

**public** **static** ArrayList<Merke> getMerkeOversikt() {

**return** *merkeoversikt*;

}

**public** String getNavn() {

**return** navn;

}

**public** **void** setNavn(String navn) {

**this**.navn = navn;

}

@Override

**public** String toString() {

**return** navn;

}

}

**Modell.java**

/\*\*

\* En bilmodell

\* **@author** Per

\*

\*/

**public** **class** Modell {

// alle modeller i systemet

**private** **static** ArrayList<Modell> *modelloversikt* =

**new** ArrayList<>();

**private** Merke merke;

**private** String navn;

/\*\*

\* constructor

\* **@param** merke - merke til denne modellen feks -> Volvo

\* **@param** navn - navnet på modellen feks -> XC90

\*/

**public** Modell(Merke merke, String navn) {

**this**.merke = merke;

**this**.navn = navn;

// legger til modellen i oversikten

*modelloversikt*.add(**this**);

}

// get og set

**public** Merke getMerke() { **return** merke; }

**public** **void** setMerke(Merke merke) {

**this**.merke = merke;

}

**public** String getNavn() { **return** navn; }

**public** **void** setNavn(String navn) {

**this**.navn = navn;

}

**public** **static** ArrayList<Modell> getModelloversikt() {

**return** *modelloversikt*;

}

@Override

**public** String toString(){

**return** merke + " " + navn;

}

}

**Kunde.java**

/\*\*

\* En kunde som har kjøpt eller leid en bil

\* **@author** Per

\*

\*/

**public** **class** Kunde {

// alle kunder

**private** **static** ArrayList<Kunde> *kunderegister* = **new** ArrayList<>();

**private** String fornavn;

**private** String etternavn;

**private** String adresse;

**private** **int** postnummer;

**private** String poststed;

/\*\*

\* constructor

\* **@param** fornavn

\* **@param** etternavn

\* **@param** adresse

\* **@param** postnummer

\* **@param** poststed

\*/

**public** Kunde(String fornavn, String etternavn, String adresse,

**int** postnummer, String poststed) {

**this**.fornavn = fornavn;

**this**.etternavn = etternavn;

**this**.adresse = adresse;

**this**.postnummer = postnummer;

**this**.poststed = poststed;

// legg til kunde i register

*kunderegister*.add(**this**);

}

// get og set

**public** String getFornavn() { **return** fornavn; }

**public** **void** setFornavn(String fornavn) {

**this**.fornavn = fornavn;

}

**public** String getEtternavn() { **return** etternavn; }

**public** **void** setEtternavn(String etternavn) {

**this**.etternavn = etternavn;

Fortsetter på neste side

**public** String getAdresse() { **return** adresse; }

**public** **void** setAdresse(String adresse) {

**this**.adresse = adresse;

}

**public** **int** getPostnummer() { **return** postnummer; }

**public** **void** setPostnummer(**int** postnummer) {

**this**.postnummer = postnummer;

}

**public** String getPoststed() { **return** poststed; }

**public** **void** setPoststed(String poststed) {

**this**.poststed = poststed;

}

**public** **static** ArrayList<Kunde> getKunderegister() {

**return** *kunderegister*;

}

**public** String toString(){

**return** fornavn + " " + etternavn;

}

}

**Transaksjon.java**

/\*\*

\* En transaksjon , leie eller salg

\* **@author** Per

\*

\*/

**public** **abstract** **class** Transaksjon {

// alle transaksjoner

**private** **static** ArrayList<Transaksjon> *transaksjonshistorikk* =

**new** ArrayList<>();

**private** Bil bil;

**private** Kunde kunde;

/\*\*

\* **@param** bil - bilen som selges eller leies ut

\* **@param** kunde - kunden som kjøper eller leier

\*/

**public** Transaksjon(Bil bil, Kunde kunde) {

**this**.bil = bil;

**this**.kunde = kunde;

// legger til i transaksjonshistorikken

*transaksjonshistorikk*.add(**this**);

}

// get og set

**public** Bil getBil() {

**return** bil;

}

**public** **void** setBil(Bil bil) {

**this**.bil = bil;

}

**public** Kunde getKunde() {

**return** kunde;

}

**public** **void** setKunde(Kunde kunde) {

**this**.kunde = kunde;

}

**public** **static** ArrayList<Transaksjon> getTransaksjonshistorikk() {

**return** *transaksjonshistorikk*;

}

}

**Salg.java**

/\*\*

\* En salgstransaksjon

\* **@author** Per

\*

\*/

**public** **class** Salg **extends** Transaksjon {

// Kommentar A

**public** **static** **final** **double** *NYREGISTRERING* = 13798.00;

// Kommentar B

**private** **static** **double**[] *omregistreringstabell* = **new** **double**[]

{ 10329.00, 7550.00, 4977.00, 1535.00 };

**private** **double** salgssum;

/\*\*

\* constructor

\* **@param** bil - bilen som er solgt

\* **@param** kunde - kunden som har kjøpt bilen

\* **@param** salgssum - salgsum inkludert registrering

\*/

**public** Salg(Bil bil, Kunde kunde, **double** salgssum) {

**super**(bil, kunde);

**this**.salgssum = salgssum;

}

// get og set

**public** **double** getSalgssum() {

**return** salgssum;

}

**public** **void** setSalgssum(**double** salgssum) {

**this**.salgssum = salgssum;

}

Fortsetter på neste side

// Kommentar C

**public** **static** Double getOmregAvgift(Bil bil) {

// Kommentar D

**if** (bil **instanceof** Nybil) {

// Kommentar E kom med forslag til feilmelding

System.*out*.println("Skriv feilmelding her");

**return** **null**;

} **else** {

// Kommentar F

**int** alder = Calendar.*getInstance*().get(Calendar.*YEAR*)

- bil.getAarsmodell();

**if** (alder == 2)

**return** *omregistreringstabell*[0];

**else** **if** (alder == 3)

**return** *omregistreringstabell*[1];

**else** **if** (alder > 2 && alder < 12)

**return** *omregistreringstabell*[2];

**else**

**return** *omregistreringstabell*[3];

}

}

}

**Utleie.java**

/\*\*

\* En transaksjon for utleie av bil

\* **@author** Per

\*

\*/

**public** **class** Utleie **extends** Transaksjon{

**private** **double** utleieprisDøgn;

**private** **int** antallUtleiedager;

/\*\*

\* constructor

\* **@param** bil - bilen som er leid ut

\* **@param** kunde - kunden som har leid bilen

\* **@param** utleieprisDøgn - prisen for å leie bilen i ett døgn

\* **@param** antallUtleiedager - antall dager bilen er utleid

\*/

**public** Utleie(Bil bil, Kunde kunde, **double** utleieprisDøgn,

**int** antallUtleiedager) {

**super**(bil, kunde);

**this**.utleieprisDøgn = utleieprisDøgn;

**this**.antallUtleiedager = antallUtleiedager;

}

// get og set

**public** **double** getUtleieprisDøgn() {

**return** utleieprisDøgn;

}

**public** **void** setUtleieprisDøgn(**double** utleieprisDøgn) {

**this**.utleieprisDøgn = utleieprisDøgn;

}

**public** **int** getAntallUtleiedager() {

**return** antallUtleiedager;

}

**public** **void** setAntallUtleiedager(**int** antallUtleiedager) {

**this**.antallUtleiedager = antallUtleiedager;

}

}

**Main.java**

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Merke audi = **new** Merke("Audi");

Modell a4 = **new** Modell(audi, "A4");

Nybil audia4 = **new** Nybil(a4, 2014, 56, 298456);

Merke bmw = **new** Merke("BMW");

Modell i320 = **new** Modell(bmw, "320i");

Bruktbil bmw320i = **new** Bruktbil(i320, 2000, 56789, 234000

,2);

Kunde per = **new** Kunde("Per", "Bisseberg", "Parkveien 8"

, 1767, "Halden");

bmw320i.leiut(per, 555, 4);

bmw320i.selg(per, 200000);

audia4.selg(per, 290000);

*skrivUtTransaksjoner*();

*skrivUtVarelager*();

}

// skriv ut alle transaksjoner

**private** **static** **void** skrivUtTransaksjoner() {

**for** (Transaksjon t : Transaksjon.*getTransaksjonshistorikk*()) {

System.*out*.println(t);

}

}

// skriv ut alle biler i lageret

**private** **static** **void** skrivUtVarelager() {

}

}