Høgskolen i Østfold

NY/UTSATT EKSAMEN

Emnekode: ITF10611	Emne: Objektorientert Programmering
Dato: 6. januar 2015	Eksamenstid: kl 09:00 til kl 13:00
Hjelpemidler: To A4-ark (fire sider) med egne notater	Faglærer: Per Bisseberg

Eksamensoppgaven:
Oppgavesettet består av 8 sider inklusiv denne forsiden. Du er selv ansvarlig for å kontrollere at oppgaven er komplett før du begynner å besvare spørsmålene. 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.

Lykke til!

Sensurdato: 27 januar 2016
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 arv.
b) Lag en illustrasjon som viser arv.
c) Lag enkel kode som viser arv.

Oppgave 1.2

Forklar disse nøkkelordene fra Java

a) static
b) implements
c) abstract

Oppgave 1.3

Forklar og eksemplifiser med kode begrepet polymorfi.

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 int x;

	public A(int x){
		x = x;
	}
}
```

Oppgave 2.1.2

```
public interface Oppg2 {
	String y = "Y";

	public String returnY(){
		return y;
	}
}
```

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).setFornavn("Anni");

	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 Elisabeth", "Hansen", 29));

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

		@Override
		public int compare(Person p1, Person p2) {
			return -(p2.getFornavn().length() - p1.getFornavn().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> genererUtskrift(String tegn){
		ArrayList<Linje> ret = new ArrayList<Linje>();
		for(Linje l: fillinjer){
			String[] mid = l.getTekst().split(tegn);
			String nyTekst = "<li>";
			for(int i = 0; i < mid.length; i++){
				if(i+1 != mid.length){
					nyTekst += mid[i] + " ";
				}
				else{
					nyTekst += mid[i] + "</li>";
				}
			}
			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"));

	System.out.println("<ul>");

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

	System.out.println("</ul>");

}
```

Oppgave 3(50%)

Du har fått i oppdrag å utvikle en prototype bestående av kjernefunksjonalitet og en grunnleggende oop-struktur for et utlånssystem for en ungdomsskole. Det som lånes ut er tekniske duppeditter som kamera, bærbare PCer, tablet/pad, mobiltelefoner, projektorer etc. De ulike enhetene har en fast plass på et lager, plassen identifiseres med en lokasjonsID bestående av reolbokstav og hyllenummer på denne måten: RAH02 som betyr: Reol A hylle nummer 2.

Vi skal holde en oversikt over hvilke personer som har lånt hvilke enheter. Disse personene er ansatte og elever ved skolen.

Følgende regler gjelder for systemet

- Enhet
    - Skal inneholde følgende verdier:
        - Id ett heltall
        - Merke
        - Modellbetegnelse
        - Hvor den befinner seg på lageret
    - Enheten skal ikke kunne lånes ut om den allerede er utlånt... doh..
    - Ulike enheter kan ha ulike maksimale antall utlånsdager. For eksempel kan et kamera lånes i 14 dager mens en mobiltelefon kan maksimalt lånes i 1 dag.
- Person
    - Skal inneholde følgende verdier:
        - Fornavn
        - Etternavn
        - En id, her benytter vi fødsels- og personnummer. Så får vi ta evt. trøbbel ang. personvern senere.
        - Telefonnummer
    - Vi må skille på om personen er ansatt eller elev fordi en ansatt kan låne flere enheter på en gang mens elever kan bare låne en ting av gangen..
    - Elver kan dessuten kun ha ett lån av gangen, det vil si at de kan ikke låne noe nytt før de har levert tilbake det de allerede har lånt fra før. Ansatte har ingen slike begrensninger.
- Utlån
    - Et utlån inneholde følgende verdier
        - Personen som har lånt enheten.
        - Enheten som er utlånt
        - Tid og dato for utlån.

Oppgave 3.1

Lag et objektorientert design (diagram) for utlånssystemet. Besvarelsen skal inneholde en grov objektmodell med alle assosiasjoner/relasjoner. Det er viktig at det kommer klart frem av diagrammet om du benytter abstrakte klasser og/eller interface og at disse representeres korrekt.

Du skal ikke modellere reoler, hyller og lignende siden du forutsetter at lokasjonsIDen er tilstrekkelig i denne sammenheng. Du skal også se bort fra eventuelle menysystemer, vindussystemer og lignende, altså ingen GUI-elementer.

Om du ser for deg nødvendigheten av å benytte container-klasser i systemet så trenger disse ikke å vises i modellen.

Oppgave 3.2

Lag kode for klasser og evt. interface du kom frem til i oppgave 3.1. Generelle Get og Set metoder trenger dere ikke ta med, med mindre du er avhengig av de for å utføre tester.

Noen naturlige klasser ville være (men det er din egen vurdering gjort i oppgave 3.1 som skal ligge til grunn):

- Utlån
    - Klasse som representerer et utlån av en enhet til en person
- Enhet
- Person
- Ansatt
- Elev