Høgskolen i Østfold

EKSAMEN

Emnekode: ITF10611	**Emne:** Objektorientert Programmering
Dato: 3 januar 2018	**Eksamenstid:** kl. 09:00 til kl. 13:00
Hjelpemidler: To A4-ark (fire sider) med egne notater	**Faglærer:** Per Bisseberg

Eksamensoppgaven:
Oppgavesettet består av 17 sider inklusiv denne forsiden og vedlegg. 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 presist.

Om en oppgave virker å forutsette at du har løst en tidligere oppgave så kan du forutsette at denne er løst uavhengig om du faktisk har gjort det eller ei. F.eks.: om du i en oppgave må benytte metoden***: metode()*** som du skulle ha implementert i en tidligere oppgave, så kan du bare anta at dette er gjort og den fungerer slik det er tenkt.

Pass også på å svare på alle oppgaver. Dersom du står fast er det 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.

Lykke til!

Sensurdato: 24 januar 2018

Karakterene er tilgjengelige for studenter i Studentweb.

Oppgave 1 (25 %)

Disse oppgavene skal besvares kort og presist. 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 konstruktør og eksemplifiser med enkel kode.
b) Forklar begrepet polymorfi/polymorfisme og eksemplifiser med enkel kode.

Oppgave 1.2

Forklar disse nøkkelordene fra Java

a) static
b) abstract
c) final

Oppgave 1.3

Gjør rede for begrepene aggregering og komposisjon. Hva er det? Hvordan benyttes det? Hvorfor benytter vi det? Hva er effektene av det? Hva skiller disse? Underbygg utredningen med kode og UML.

Oppgave 2 (25 %)

Analyse av kode.

Oppgave 2.1 (15 %)

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 abstract class A {
     private String kode;
}

public class B extends A{
     private String navn;

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

Oppgave 2.1.2

```
public abstract class  A {

     public abstract void formatPrint() {
          System.out.println(navn + ": " + verdi);
     };
}

public class B extends A{
     private String navn;
     private int verdi;

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

Oppgave 2.1.3

```
public final class A {

     private String navn;

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

public class B extends A{
     public String kode;

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

Oppgave 2.2 (10 %)

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

Oppgave 2.2.1

```
public class MC {
	private String merke;
	private String modell;
	private String aarsmodell;

	public MC(String merke, String model, String aarsmodell) {
		this.merke = merke;
		this.modell = model;
		this.aarsmodell = aarsmodell;
	}

	@Override
	public String toString() {
		return aarsmodell + " - " + merke + " '" + modell + "'";
	}

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

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

	mcer.add(new MC("Harley Davidson", "Electra Glide", "2017"));
	mcer.add(new MC("Yamaha", "YZF R15", "2015"));
	mcer.add(new MC("BMW", "K 1600 B", "2016"));


	for(MC mc : mcer) {
		System.out.println(mc);
	}

}
```

Oppgave 2.2.2

Vi benytter MC-klassen fra forrige oppgave

```
public static void main(String[] args) {
	ArrayList<MC> mcer = new ArrayList<>();

	mcer.add(new MC("Harley Davidson", "Electra Glide", "2017"));

	mcer.add(new MC("Yamaha", "YZF R15", "2015"));
	mcer.add(new MC("BMW", "K 1600 B", "2016"));

	Collections.sort(mcer, new Comparator<MC>() {

		@Override
		public int compare(MC mc1, MC mc2) {
			return mc1.modell.length() - mc2.modell.length();
		}

	});


	for(MC mc : mcer) {
		System.out.println(mc);
	}
}
```

Oppgave 3 (50 %)

Vi skal utvikle en prototype av et system for en oppdragsgiver, en bemidlet hipster. Vi har nettopp begynt utviklingen og vi jobber med å få på plass de første delene av kjernefunksjonaliteten. Vi trenger ikke tenke på persistent lagring på dette tidspunktet, du kan anta at dette allerede er på plass slik at systemet til enhver tid vil holde på lagret informasjon. Vedlagt finner du de klasser vi har så langt.

Hipsteren «Tore Flanellbart» trenger et system som holder oversikt over hans omfangsrike vinylsamling. Han ønsker at dette skal utvikles som en desktop-applikasjon i Java, han vil ABSOLUTT ikke ha en moderne webløsning.

Prototypen må muliggjøre:

- Registrering av utgivelser med differensiering av album og singler
- Registrering av spor («sanger») og på hvilke utgivelser vi finner disse.
- Registrering av musikere som er medvirkende på et spor og hvilket instrument de spiller på sporet
- Registrering av artister med differensiering av soloartister og band

Elementene i systemet er som følger:

- **Main**
  - **Denne klassen viser et typisk scenario systemet skal støtte, les denne og prøv å få en oversikt om hvordan det henger sammen.**
- Utgivelse
  - Registreres med en unik kode (ISMN), navn, artist, utgivelsesdato, hvilket plateselskap som står for utgivelsen og sjanger.
- LP
  - En utgivelse i form av et album.
- SP
  - En utgivelse i form av en singel. Det er normalt at en singel KAN «tilhøre» et album, om det er tilfelle må dette registreres.
- Innspilling
  - En innspilling er et unikt opptak av en «sang/låt». En innspilling registreres med en unik kode (ISMN), navn, lengde i sekunder. I tillegg kan en innspilling forekomme på flere utgivelser og ha en rekke utøvende musikere.
  - MusikerInstrument
    - En indre klasse i Innspilling som registrerer hvilke musikere og instrument de har benyttet i innspillingen.
- Musiker
  - En musiker som registreres med personalia. Vi regner også vokalister som musikere.
- Artist
  - En klasse som benyttes for å definere hovedartistene(e) til en utgivelse.
- Solo
  - En soloartist er en musiker.
- Band
  - Et band er registrert kun med bandnavn. Man kunne tenkt seg at et band var en samling musikere, men på dette tidspunktet i utviklingen nøyer vi oss med kun bandnavn.
- Plateselskap
  - Et plateselskap, registreres kun med et navn.
- Sjanger
  - En sjanger, registreres kun med navn, f.eks. «Jazz», «Prog Rock» etc.

Oppgave 3.1

Lag et klassediagram, det er ikke nødvendig å ta med felter og metoder – **kun klassenavn/interfacenavn**, over alle elementene i systemet hvor du tydelig får frem eventuelle arverelasjoner, implementasjoner og ulike typer assosiasjoner. Du trenger ikke tegne inn klassen Main. Du trenger heller ikke å tegne nye elementer eller assosiasjoner som evt. dukker opp i systemet i senere oppgaver.

Oppgave 3.2

Skriv ferdig konstruktørene i klassene **LP** og **SP.** NB: **SP** kan holde en referanse til et album (**LP**), du må her lage konstruktører som støtter begge muligheter.

Oppgave 3.3

I klassene Solo og Band skal du implementere den abstrakte metoden definert i klassen Artist. Det er ønskelig å få ut henholdsvis bandnavn og soloartistens for- og etternavn.

Oppgave 3.4

I klassen **Innspilling** finner du flere uferdige metoder. Du skal fullføre logikken i disse.

a) **leggTilMusikerInstrument:**
   Skriv ferdig logikken som må til for å legge til et nytt MusikerInstrument-objekt til innspillingen.
b) **hentInnspillingMedMusiker**:
   Skriv ferdig logikken som må til for å hente alle innspillinger en gitt musiker har vært medvirkende på.
c) **hentInnspillingMedMusikerInstrument**:
   Skriv ferdig logikken som må til for å hente alle innspillinger en gitt musiker har vært medvirkende på med et gitt instrument.

Oppgave 3.5

Gjør tillegg/forandring i klassen **Utgivelse** som definerer at den naturlige sorterings-ordenen på Utgivelsessobjekt er stigende på **utgivelsesdato**.

Oppgave 3.6

Vi har fått et ønske fra Tore Flanellbart om å legge til funksjonalitet som lar han registrere hvor utgivelsene befinner seg fysisk. Han forklarer at han har dedikerte og merkede hyller på 3 lokasjoner (Hjemme, Hytta og Kontoret), hyllene er heltallsnummerert (fra og med 1) på hver lokasjon. Hjemme har han 42 hyller (1 - 42), på hytta har han 12 hyller (1 – 12) og på kontoret har han 3 hyller (1 - 3).

Han ønsker å kunne finne en utgivelse i systemet og se hvor den fysisk befinner seg og i hvilken hylle.

Du skal gi han denne nye funksjonaliteten.

Gjør rede for hvordan du ville løst dette ved å:

- Benytte UML (du trenger ikke tegne hele systemet på nytt, kun de delene som påvirkes av omstruktureringen)
- Skrive kode for nye systemelementer
- Oppdatere eksisterende kode

Vedlegg til oppgave 3

Klasser og Interface:

- Main – Trenger ikke være med i oppgave 3.1 UML
- Utgivelse
- LP
- SP
- Innspilling
    - MusikerInstrument
- Musiker
- Artist
- Solo
- Band
- Plateselskap
- Sjanger

Vedlegget er på 7 sider inklusive denne siden.

Main

```
public class Main {

	public static void main(String[] args) {

		// Tom Waits
		Musiker tomWaits = new Musiker("Tom", "Waits", "USA",
			new GregorianCalendar(1949, 11, 7), true);

		// plateselskap for utgivelsen "Swordfishtrombones"
		Plateselskap island = new Plateselskap("Island Records");

		// sette tomWaits som soloartist
		Solo tomWaitsSolo = new Solo(tomWaits);

		// Registrerer albumet "Swordfishtrombones"
		LP swordfishtrombones = new LP(
			"979-0-2600-0016-7",
			"Swordfishtrombones", tomWaitsSolo, island,
			new GregorianCalendar(1983, 8, 1),
			new Sjanger("Experimental Rock")
		);

		// legger til bare 1 innspilling i albumet så det ikke blir
		uoversiktelig)
		Spor franks = new Spor("979-0-1145-1256-1","Frank's Wild Years",110);

		// legger til musikere på denne innspillingen
		// lager bare noen av de pga at det skal være oversiktelig

		// lager musikere først
		Musiker jayAnderson = new Musiker("Jay", "Anderson", "Canada",
			new GregorianCalendar(1955, 9, 24), true);

		Musiker morisTepper = new Musiker("Moris", "Tepper", "USA",
			new GregorianCalendar(1952,1,23), true);

		// legger til musikere på innspilling med instrument
		franks.leggTilMusikerInstrument(jayAnderson, "Bass");
		franks.leggTilMusikerInstrument(morisTepper, "Guitar");
		franks.leggTilMusikerInstrument(tomWaits, "Vocals");

		// legger til inspillingen i albumet (LPen)
		franks.leggTilIutgivelse(swordfishtrombones);;

	}

}
```

Utgivelse

```
public abstract class Utgivelse {

	private String ISMN;
	private String navn;
	private Artist artist;
	private Plateselskap plateselskap;
	private GregorianCalendar utgivelsesdato;
	private Sjanger sjanger;

	// liste som holder på alle produksjoner i systemet
	public static final ArrayList<Utgivelse> utgivelser = new ArrayList<>();

	// konstruktør
	public Utgivelse(String ISMN, String navn, Artist artist,
		Plateselskap plateselskap, GregorianCalendar utgivelsesdato,
		Sjanger sjanger) {

		this.ISMN = ISMN;
		this.navn = navn;
		this.artist = artist;
		this.plateselskap = plateselskap;
		this.utgivelsesdato = utgivelsesdato;
		this.sjanger = sjanger;
		utgivelser.add(this);
	}

	// Tekstlig representasjon av et Utgivelse-objekt
	@Override
	public String toString() {
		return utgivelsesdato.get(Calendar.YEAR) + ": " + navn + " - " + artist;
	}


	// metode som henter alle produksjoner for en "Artist" (band eller soloartist)
	public static ArrayList<Utgivelse> hentUtgivelserForArtist(Artist artist) {

		// Lager en liste for returnering
		ArrayList<Utgivelse> returnliste = new ArrayList<>();

		// går igjennom alle utgivelsene
		for (Utgivelse utgivelse: utgivelser) {
			// om utgivelsen har denne artisten så legges den i listen
			// for returnering
			if(utgivelse.artist.equals(artist)) {
				returnliste.add(utgivelse);
			}
		}

		return returnliste;

	}
}
```

LP

```
public class LP extends Utgivelse{

	// Oppgave 3.2

}
```

SP

```
public class SP extends Utgivelse{

	// en singel (SP) kan tilhøre et album (LP)
	private LP tilhørerAlbum;

	// Oppgave 3.2

}
```

Innspilling

```
public class Innspilling {

	private String ISMN;
	private String navn;
	private int lengde;

	// liste over hvilke produksjoner (Album eller singler denne innspillingen er utgitt på)
	private ArrayList<Utgivelse> utgivelser = new ArrayList<>();

	// liste over musikere som er medvirkende på innspillingen
	private ArrayList<MusikerInstrument> musikere = new ArrayList<>();

	// liste over alle innspillinger registrert i systemet
	private static final ArrayList<Innspilling> innspillinger = new ArrayList<>();

	// konstruktør
	public Spor(String ISMN, String navn, int lengde) {
		this.ISMN = ISMN;
		this.navn = navn;
		this.lengde = lengde;
		innspillinger.add(this);
	}

	// metode for å legge til produksjon denne innspillingen er utgitt på
	public void leggTilIutgivelse(Utgivelse utgivelse) {
		utgivelser.add(utgivelse);
	}

	// metode for å hente alle innsppillinger med en gitt musiker
	public ArrayList<Innspilling> hentInnspillingMedMusiker(Musiker musiker) {
		//Oppgave 3.3b

	}

	// metode for å hente alle innspillinger med en gitt musiker på et gitt instrument
	public ArrayList<Innspilling> hentInnspillingMedMusikerInstrument(Musiker musiker,
		String instrument) {
		// Oppgave 3.3c

	}

	// Metode for å legge til MusikerInstrument-objekt til dette sporet
	public void leggTilMusikerInstrument(Musiker musiker, String instrument) {
		// oppgave 3.3a

	}

	// indre klasse for musiker og hvilket instrument denne spilte på denne
	// innspillingen.
	// For å begrense kompleksiteten er et instrument en tekststreng og ikke en klasse.
	private class MusikerInstrument {
		private Musiker musiker;
		private String instrument;

		// Konstruktør
		public MusikerInstrument(Musiker musiker, String instrument) {
			this.musiker = musiker;
			this.instrument = instrument;
			musikere.add(this);
		}
	}
}
```

Musiker

```
public class Musiker {

	private String fornavn;
	private String etternavn;
	private String nasjonalitet;
	private GregorianCalendar fødselsdato;
	private GregorianCalendar dødsdato;

	// denne tar kun høyde for at artister er enten kvinne eller mann.
	// Det burde vært andre muligheter her,men det gjøres slik pga kompleksitet i koden.
	private boolean erMann;

	// konstruktør - levende artist
	public Musiker(String fornavn, String etternavn, String nasjonalitet,
		GregorianCalendar fødselsdato, boolean erMann) {

		this.fornavn = fornavn;
		this.etternavn = etternavn;
		this.nasjonalitet = nasjonalitet;
		this.fødselsdato = fødselsdato;
		this.erMann = erMann;
	}


	// konstruktør - død artist
	public Musiker(String fornavn, String etternavn, String nasjonalitet,
		GregorianCalendar fødselsdato, GregorianCalendar dødsdato, boolean erMann) {

		this.fornavn = fornavn;
		this.etternavn = etternavn;
		this.nasjonalitet = nasjonalitet;
		this.fødselsdato = fødselsdato;
		this.dødsdato = dødsdato;
		this.erMann = erMann;
	}

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

Artist

```
public abstract class Artist {

	@Override
	public abstract String toString();

}
```

Solo

```
public class Solo extends Artist{

	// en soloartist er en "Musiker"
	private Musiker artist;

	public Solo(Musiker artist) {
		this.artist = artist;
	}

	// Oppgave 3.3

}
```

Band

```
public class Band extends Artist{

	// bandnavn som streng
	private String navn;

	public Band(String navn) {
		this.navn = navn;
	}

	// Oppgave 3.3

}
```

Plateselskap

```
public class Plateselskap {

	private String navn;

	public Plateselskap(String navn) {
		this.navn = navn;
	}
}
```

Sjanger

```
public class Sjanger {

	private String navn;

	public Sjanger(String navn) {
		this.navn = navn;
	}
}
```