Seminaropplegg, uke 4

- Repetisjon - Arv og subklasser
- Polymorfi
- Oppgaver
    - Diskusjon

Arv og subklasser - repetisjon

Enkel repetisjon av arv og subklasser fra forrige uke. Nevn viktige begreper som:

- extends
- instanceof

toString og equals

Nevn at metodene toString() og equals ligger i object og kan redefineres av alle klasser. Utvid gjerne Kvadrat/Rektangel-eksempelet under med toString og equals-metoder.

Polymorfi

Klassen *Object*
Nevn klassen object, og at alle klasser i java er en subklasse av denne klassen.

Kontruktører i subklasser og nøkkelbegrepet super

Forklar at man overskriver konstruktøren til en superklasse ved å opprette en ny konstruktør i subklassen. Begynn gjerne på et eksempel med firkant som superklasse og boks som subklasse. Forklar begrepet super, og at dette brukes for å gjøre et kall på superklassens konstruktør. Husk at dette må være det første som gjøres i subklassens konstruktør(er).

```
public class Rektangel {
    protected int lengde;
    private int bredde;

    public Rektangel(int lengde, int bredde) {
      this.lengde = lengde;
      this.bredde = bredde;
    }
}

public class Kvadrat extends Rektangel {

    public Kvadrat(int lengde) {
      super(lengde, lengde);
    }
}
```

Omdefinering av metoder - Overriding

Når en subklasse redefinerer en metode definert i en av subklassene. Fortsett gjerne på Firkant-Boks eksempelet, hvor du kan lage en metode areal som returnerer arealet til figuren. Forklar at metoder blir overskrevet .

Notis: Etter å ha endret til Kvadrat extends Rektangel, er det ikke nødvendig å overskrive areal. Løs gjerne oppgaven etter eksempelet under, og la deretter studentene finne ut av hvordan eksempelet kan forbedres.

```
public class Rektangel {
    // … kode fra forrige eksempel

    public int areal() {
        return lengde * bredde;
    }
}


public class Kvadrat extends Rektangel {
    // … kode fra forrige eksempel

    @override
    public int areal() {
        return lengde * lengde;
    }
}
```

Overloading

Metoder med samme metodenavn, men forskjellig *signatur*.

Metode	**Signatur**
public class settDimensjoner(int hoyde)	settDimensjoner(int)
public class settDimensjoner(int hoyde, int bredde)	settDimensjoner(int, int)

Trix-oppgaver:

Nevn at det er mulig å sende inn forslag til Trix-oppgaver og at studenter som deltar vil bli kreditert med navn (hvis de ønsker det). Foreslå at studentene jobber sammen i grupper på 2-3 hvis de vil gjøre dette.

Hvis det passer for din gruppe kan studentene gjerne jobbe i grupper under seminaropplegget for å gjøre dette. Et viktig punkt er at oppgaven må være relevant til pensumet vi skal gjennom i kurset).

Oppgaver sendes til siljemda@ifi.uio.no sammen med et notat med stikkord om hvilke temaer oppgaven dekker.

Oppgaver

1.a - Dyr, Hund og Katt:
Skriv en klasse Dyr. Klassen skal være tom, utenom en metode lagLyd(). Lag deretter to subklasser av dyr; Hund og Katt.

Om man kaller på lagLyd i et hundeobjekt skal det printes *"Voff!"* til terminalen. Om man kaller på lagLyd i et katteobjekt skal det printes "Mjau!"

```
abstract class Dyr{
	public abstract void lagLyd();
}

class Hund extends Dyr{

	public void lagLyd(){
		System.out.println("Voff!");
	}
}

class Katt extends Dyr{

	public void lagLyd(){
		System.out.println("Mjau!");
	}
}
```

1.b - Diskuter i grupper: Hvilken fordel får vi ved å ha en superklasse med en tom metode lagLyd?

I felleskap/gruppen: Skriv et testprogram som demonstrerer denne fordelen.

```
public class Test{
	public static void main(String[] args)
{

		Dyr dyr1 = new Hund();
		Dyr dyr2 = new Katt();

		dyr1.lagLyd();
		dyr2.lagLyd();

	}
}
```

1.c - Tegning:
På egen hånd: Tegn klassehierarki og

datastruktur til oppgaven. Diskuter med gruppen: Sammenlign tegninger. Hva har dere gjort ulikt?	

2.a - Person og student:
Diskuter hvilke metoder her som er eksempler på overriding, og hvilke metoder som er eksempler på overloading. Fyll inn @Override der det hører hjemme.
 	
I klassen Person:
registerPerson(String) og registrerPerson(String, String) > **overloading**.
 	
I klassen Student:
registrerPerson(String, String) og registrerPerson(String, String,String) > **overrider** klassen Person sine registrerPerson-metoder.
 	
registrerPerson(String, String) og registrerPerson(String, String,String) > **overloading**.
 	
skrivPerson() i Student **overrider** klassen Person sin skrivPerson().
	Kode:
	`class Person {`
	`  String navn = "ikke registrert.";`
	`  String adresse = "ikke registrert.";`
	
	`  public void registrerPerson(String n) {`
	`  navn = n;`
	`  }`
	
	`  public void registrerPerson(String n, String a) {`
	`  navn = n;`
	`  adresse = a;`
	`  }`
	
	`  public void skrivPerson(){`
	`  System.out.println("Navn: " + navn + ", adresse: " + adresse);`
	`  }`
	`}`
	
	`class Student extends Person {`
	`  String idnr = "ikke registrert.";`
	
	`  public void registrerPerson(String n, String i) {`
	`  navn = n;`
	`  idnr = i;`
	`  }`
	
	`  public void registrerPerson(String n, String i, String a) {`
	`  navn = n;`
	`  idnr = i;`
	`  adresse = a;`
	`  }`
	
	`  public void skrivPerson(){`
	`  System.out.println("Navn: " + navn + ", studentnr: " + idnr + ", adresse: " + adresse);`

	} }

2.b - Opprettelse av person og studenter: Opprett 3 personer. 1 av dem skal være en student. Velg selv hvilke metoder du vil benytte, men prøv ut litt forskjellig. Skriv deretter ned hva som skrives ut hvis du kaller på “skrivPerson()” på Person-objektene dine. *Diskusjon: hva skrives ut her:* Person a = new Student(); a.registrerPerson(“Lise”, “Holmveien 2”); a.skrivPerson(); *Diskusjon 2:* *Hva skrives ut her:* Person a = new Student(); a.registrerPerson(“Siri”); a.skrivPerson(); *Diskusjon 3:* *Hva skjer her:* Person a = new Person(); a.registrerPerson(“Siri”, “1235”, “Trimveien 9”); a.skrivPerson();	