## Seminaropplegg uke 9

**Tema:**

* Design og testing

**Design:**

Hva er et godt design? (fra IN1030):

* En god utforming gjør den jobben den er ment å gjøre.m
* En god utforming er enkel og elegant.
  + Eleganse innebærer å finne akkurat riktig abstraksjonsnivå.
* En god utforming er gjenbrukbar, utvidbar og enkel å forstå.
* Et godt objekt har et lite og veldefinert ansvarsområde.
* Et godt objekt skjuler implementasjonsdetaljer fra andre objekter

**Testing**

Ha som mål å knekke programmet, ikke å få alle testene til å kjøre. lurt Det kan være å skrive testene før selve programmet, da dette gjør at du vil måtte tenke på eventuelle utfordringer før man begynner med kodingen - lite gjennomtenkt kode fører fort til spagettikode.

Vis hvordan man kan skrive enkle tester som kan brukes til å “automatisere” obligtesting. Dette er også oppgave 1a, men det er nok lurt å repetere hvorfor det går an å ha to metoder med samme navn, men med forskjellige parametere.

|  |
| --- |
| test("liste, index 2", liste.hent(2), 12);  boolean test(String beskrivelse, int inn, int forventet){  System.out.print("TEST: " + beskrivelse + ": " + inn + " == " + forventet + ": ");  if(inn == forventet){  System.out.println("RIKTIG");  return true;  }else{  System.out.println("FEIL");  return false;  }  }  boolean test(String beskrivelse, double inn, double forventet){  System.out.print("TEST: " + beskrivelse + ": " + inn + " == " + forventet + ": ");  if(inn == forventet){  System.out.println("RIKTIG");  return true;  }else{  System.out.println("FEIL");  return false;  }  }  boolean test(String beskrivelse, String inn, String forventet){  System.out.print("TEST: " + beskrivelse + ": " + inn + " == " + forventet + ": ");  if(inn.equals(forventet)){  System.out.println("RIKTIG");  return true;  }else{  System.out.println("FEIL");  return false;  }  } |

**Oppgaver:**

Oppgavene fokusere mest på testing, og kun noe på designvalg, og derfor er oppgavene fra tidligere år lagt med nederst i dette dokumentet.

**Oppgaver finnes på GitHub:** [**https://github.uio.no/IN1010/seminaroppgaver\_v19/blob/master/uke09.md**](https://github.uio.no/IN1010/seminaroppgaver_v19/blob/master/uke09.md)

Oppgavene er store, men hovedfokuset er å lage tester og planlegge hva metodene kommer til å returnere. Prioriter gjerne de rekursive metodene når det kommer til implementasjon av metodene.

**Gamle oppgaver; se GitHub for årets oppgaver: .**

**Program til oppgavene:**

|  |
| --- |
| import java.util.Scanner;  class ReverseNumber {  public static void main(String args[]) {  int num = 0;  int reversenum = 0;  System.out.println("Tast inn et heltall, og trykk enter: ");  // Les input fra konsollet:  try (Scanner in = new Scanner(System.in)) {  // Putt lest tall i num-variabelen  num = in.nextInt();  // while loop for å reversere tallet  while (num != 0) {  reversenum = reversenum \* 10;  reversenum = reversenum + num % 10;  num = num / 10;  }  System.out.println("Input reversert er: " + reversenum);  }  }  } |

**Oppgave 1:**

|  |
| --- |
| Hvilke ansvarsområder har dette programmet?  Hvordan kan det være fornuftig å dele det opp?  Er det noen designprinsipper man bør tenke på? |

**Oppgave 2:**

|  |
| --- |
| Basert på svaret i deloppgave 1, skriv noen enkle enhetstester som sjekker at programmet faktisk reverserer tall i henhold til forventningene. Er det noe annet som man bør teste? |

**Oppgave 3:**

Diskuter gode grunner til hvorfor det er viktig med god kodedesign? Hva er god kodedesign?

**Løsning:**

Alt som er nevnt øverst under “god design”.

Generelt viktig:

Lesbarhet.

Vedlikeholdbarhet.

Gjenbrukbarhet.

Effektivitet.

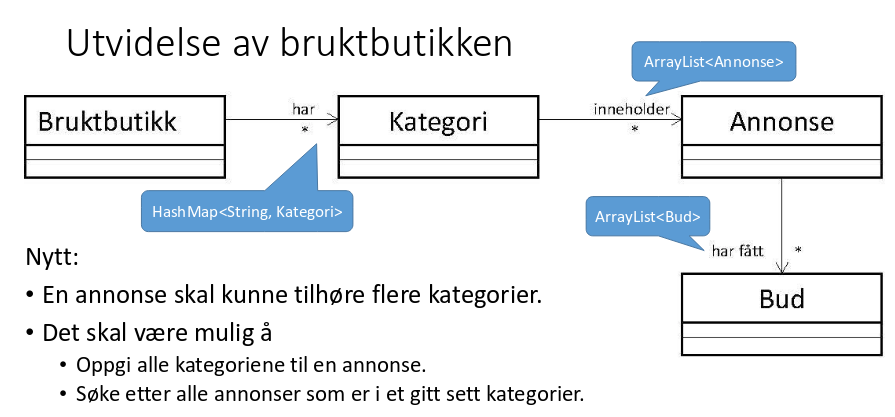
Robusthet.

Pålitelighet.

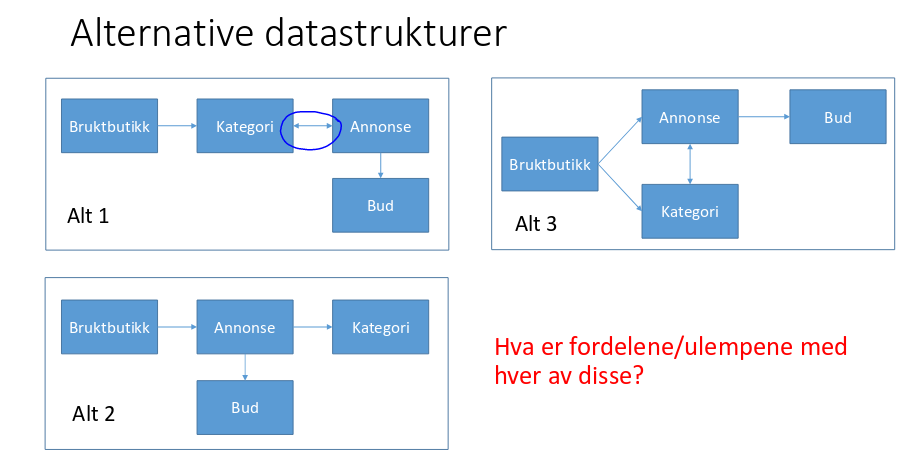
Korrekthet.

**Oppgave 4: (her må gruppelærer forklare tegningen kort siden UML ikke er undervist).**

Vi har en bruktbutikk, UML-en (som viser relasjonene mellom klassene) for programmet ser slik ut nå. Du skal legge til noen utvidelser i det allerede eksisterende programmet, de nye endringene som skal til ligger under:



Du kommer opp med følgende alternative datastrukturer som du mener kan løse de nye endringene i programmet. Diskuter fordeler og ulemper med de nye designene dine:



**Løsningsforslag:**

**Testing:**

**…..**

**Repetisjon rekursjon:**

*Moro: gå inn på Google.com og søk på “recursion”.*