HØGSKOLEN I BERGEN
Avdeling for ingeniørutdanning
Institutt for data- og realfag

EKSAMEN I : DAT100 Grunnleggende programmering

KLASSE : 1. klasse DATA/INF

DATO : 20. desember 2013

ANTALL OPPGAVER : 4
ANTALL SIDER : 9 (uten vedlegg)
VEDLEGG : 3 sider

HJELPEMIDLER : Ingen

TID : 4 timer (9.00 – 13.00)

MÅLFORM : Bokmål

SENSOR :

FAGLÆRER(E) : Sven-Olai Høyland
Pål Ellingsen
Harald Soleim

MERKNAD : Der det er gitt kodeeksempel, kan navn i koden oversettes til bokmål/nynorsk uten kommentar.

Postboks 7030, 5020 Bergen. Tlf. 55 58 75 00, Fax 55 58 77 90
Besøksadr.: Nygårdsgt. 112, Bergen

Oppgave 1 (Vekt 10% ~ 30 minutt)

I hver av deloppgavene under er det oppgitt en del av et Java-program som lar seg kompilere og kjøre. Du skal besvare hver oppgave ved å fortelle hva som blir skrevet ut når den aktuelle koden kjøres.

a)

```
public static void main(String[] args) {
      int x=4;
      System.out.println("y = ");
      System.out.println(x * 2 + (x / 3) * (x - 3));
}
```

b)

```
public static void main(String[] args) {
      for(int i = 10; i > 0; i = i - 2){
            System.out.print(i - 1);
      }
}
```

c)

```
public class Oppg1c {
      public static void main(String[] args) {
            AnnenKlasse a = new AnnenKlasse();
            AnnenKlasse b = new AnnenKlasse(2);
            System.out.println(a.regnUt());
            System.out.println(b.regnUt(3));
            System.out.println(b.regnUt(10));
      }
}

public class AnnenKlasse{
      private int tall;
      public AnnenKlasse(){
            tall = 1;
      }
      public AnnenKlasse(int b){
            tall = b * b;
      }
      public int regnUt(){
            return tall * tall;
      }
      public int regnUt(int x){
            return tall * tall * tall;
      }
}
```

d)

```
public class Oppg1d {
      public static void main(String[] args) {
            KlasseEn etObjekt = new KlasseTo();
            System.out.println(etObjekt);
      }
}

private class KlasseEn {
      public double tall(){
            return 2.0;
      }
}

public class KlasseTo extends KlasseEn {
      public double tall() {
            double d = super.tall() + 2;
            return d;
      }

      public String toString() {
           return (tall() + " ");
     }
}
```

e)

```
public class Oppg1e {
      public static void main(String[] args) {
            try{
                  int x = 3;
                  x = x / 0;
                  System.out.println(x);
            }
            catch (NullPointerException ex){
                  System.out.println("Unntak nr. 1 kastet.");
            }
            catch (ArithmeticException ex){
                  System.out.println("Unntak nr. 2 kastet.");
            }
      }
}
```

Oppgave 2 (Vekt 20% ~ 45 minutt)

a) For hvert av punktene under skal du avgjøre om det vil være passende for den ene av de to klassene å være en subklasse av den andre. Hvis så, si hvilken klasse som skal være subklasse. Hvis ikke, forklar hvorfor.

`Trekant` og `Rektangel`
`Bok` og `Bibliotek`
`Student` og `Lærer` ved en høgskole
`Lærer` og `Ansatt` ved en høgskole
`Ansatt` og `Person` ved en høgskole
`Datamaskin` og `ElektroniskUtstyr`
`Datamaskin` og `Operativsystem`
`Datamaskin` og `Prosessor` (CPU)

b) Klassen `C` er gitt i rammen under. Deklarer en klasse `D` som er en subklasse til `C` som *overkjører (omdefinerer)* metoden `m1()` slik at metoden returnerer differansen mellom `m` og `n`, `(m - n)`, i stedet for deres sum.
Utstyr klassen `D` også med en metode `m2()` som returnerer verdien av

```
(m + n) * (m - n)
```

ved å benytte de to versjonene av metoden `m1()`.

```
public class C {
      private int m, n;

      public C (int startM, int startN) {
            m = startM;
            n = startN;
      }

      public int finnM() {
            return m;
      }

      public int finnN() {
            return n;
      }

      public int m1() {
            return m + n;
      }
}
```

c) Klassen To skal brukes til å opprette objekter som inneholder et heltall og en metode for å gjøre en beregning med dette heltallet – se under.

1. Hvilke(n) feil vil kompilatoren gi melding om når To kompileres? Klassen En forutsettes feilfri og tilgjengelig.

2. Vis hvordan klassen kan rettes opp slik at kompilatoren ikke gir feilmelding

```
public abstract class En {
      abstract public int beregn();
}
```

```
public class To extends En {
      private int verdi;

      public To() {
            verdi = 0;
      }

      public To(int v) {
            verdi = v;
      }

      public int finnVerdi() {
            return verdi;
      }

      public int regnUt() {
            return verdi * 2;
      }
}
```

Oppgave 3 (Vekt 20% ~ 45 minutt)

I denne oppgaven skal du skrive Java-koden som mangler og tegne et objekt-diagram (en type UML-diagram). Programmet har fem klasser: `Bestillingsprogram`, `Bestilling`, `Servenhet`, `Mat` og `Drikke`. Det er **bare** main-metoden i `Bestillingsprogram` som ikke er ferdig skrevet. Programmet brukes til å registrere bestillinger fra gjester ved et bord i en restaurant.

Bestilling.java

```
public class Bestilling {
	private Servenhet[] bestilte;
	private int antEnheter;
	public Bestilling() {
		bestilte = new Servenhet[100];
		antEnheter = 0;
	}
	public void leggtil(Servenhet nyEnhet){
		if (antEnheter < 100){
			bestilte[antEnheter] = nyEnhet;
			antEnheter++;
		}
	}
	public String toString(){
		String tekst = "Bestilt:\n****************\n";
		for(int i=0; i<antEnheter; i++){
			tekst+=bestilte[i] + "\n";
		}
		tekst += "****************\nTotalt " + sum() + " kr.";
		return tekst;
	}
	private int sum(){
		int totsum = 0;
		for(int i = 0; i < antEnheter; i++){
			totsum += bestilte[i].getPris();
		}
		return totsum;
	}
}
```

Servenhet.java

```
public abstract class Servenhet {
	protected String navn;
	protected int pris;
	public Servenhet(String navn, int pris) {
		this.navn = navn;
		this.pris = pris;
	}
	public String getNavn() { return navn; }
	public void setNavn(String navn) { this.navn = navn; }
	public int getPris() { return pris; }
	public void setPris(int pris) { this.pris = pris; }
}
```

Mat.java

```
public class Mat extends Servenhet {
	private int gram;
	public Mat(String navn, int gram, int pris) {
		super(navn, pris);
		this.gram = gram;
	}
	public int getGram() { return gram; }
	public void setGram(int gram) { this.gram = gram; }
	public String toString() {
		return navn + ", " + gram + " gram, " + pris + " kr.";
	}
}
```

Drikke.java

```
public class Drikke extends Servenhet {
	private double liter;
	public Drikke(String navn, double liter, int pris) {
		super(navn, pris);
		this.liter = liter;
	}
	public double getLiter() { return liter; }
	public void setLiter(int liter) { this.liter = liter; }
	public String toString() {
		return navn + ", " + liter + " liter, " + pris + " kr.";
	}
}
```

Bestillingsprogram.java

```
public class Bestillingsprogram {
	public static void main(String[] args) {
		Bestilling bord1 = new Bestilling();
		// Fyll inn her
		System.out.println(bord1);
	}
}
```

a) Skriv Java-koden som mangler for klassen Bestillingsprogram slik at følgende utskrift blir skrevet til skjermen:

```
Bestilt:
****************
Pizza, 300 gram, 160 kr.
CocaCola, 0.4 liter, 36 kr.
Vann, 0.6 liter, 0 kr.
Spagetti, 150 gram, 80 kr.
****************
Totalt 276 kr.
```

b) Lag et objektdiagram for koden som ble skrevet i oppgave a).
Hvert objekt skal ha indikert hvilken datatype eller klasse den er, og alle variable skal ha indikert verdi. La også koblingen mellom objektene komme klart fram.

Oppgave 4 (Vekt 50% ~ 120 minutt)

Mange idrettslag har lagt til rette for turer i skog og mark ved å merke og rydde stier. For å motivere folk for å delta, kan de også ha lagt ut poster der folk kan registrere seg. Ofte kan en få diplom / medalje om en har vært på en post for eksempel 30 ganger i løpet av en sesong. En annen mulighet for premie er at man har vært innom alle poster i ei liste, for eksempel [«Fløyen», «Ulriken», «Løvstakken», «Lyderhorn»]. Vi skal lage deler av et system for å administrere et slikt trimopplegg.

a) Lag klassen `Post`. Klassen skal inneholde:
   - Objektvariabler (feltvariabler) som skal være private.
     - `navn` – Navnet på posten (tekststreng).
     - `merkaSti` – Om der finst merket sti til posten (sann/usann).
     - `kode` – Heltalskode, unik for hver post.
   - Konstruktør med parametre der vi kan gi verdi til alle objektvariablene.
   - Hent- og settmetodar (get/set-metoder) for alle objektvariablane.
   - `toString()` metode som returnerer en streng på forma

     "**Fløyen**, Kode: **1**, **merka sti**" eller "**Kvitebergnova**, Kode: **15**, **ikkje merka sti**".

b) Du finner API-dokumentasjon for klassene `Tur` og `Dato` vedlagt. Du skal ikke skrive kode for disse klassene, men du kan bruke metoder fra de. Lag klassen `Deltaker`. Klassen skal inneholde:

   - Objektvariabler (feltvariabler) som skal være private.
     - `navn` – navnet til deltakeren.
     - `turer` – tabell (array) med pekere (referanser) til turobjekter.
     - `antall` – antall turobjekter som faktisk er i tabellen.

   - Konstruktør **`public Deltaker(String navn)`**. Navnet på deltakeren er gitt som parameter. Konstruktøren skal opprette en tabell (array) med plass til `100` turer.

   - Public metoder
     - **`public void`** `leggTil(Tur t)`
       Objektmetode som legger til en tur i neste ledige posisjon i tabellen dersom der er plass. Dersom tabellen er full skal det skrives en feilmelding.

     - **`public int`** `antallGanger(int kode)`
       Objektmetode som returnerer hvor mange ganger en deltaker har besøkt en post gitt ved parameteren `kode`.

- `public boolean besøktAlle(int[] tab)`
  Objektmetode som sjekker om en deltaker har besøkt alle postene som er gitt som en tabell av postkoder. Metoden skal returnere true om alle postene er besøkt, false ellers.

- Lag en metode for å avgjøre om en deltaker har besøkt en spesiell post i løpet av en periode, dvs. fra og med en dato til og med en dato (for eksempel: 1. juni 2013 – 14. juni 2013). Her må du også lage første linjen selv.

c) Vi skal lage et forenklet main-program for å administrere deltakerene. Vi skal bruke en `HashMap` der vi har navnet til deltakaren som nøkkel (antar ingen har samme navn). Du finner deler av API-dokumentasjonen for `HashMap` som vedlegg. Programmet skal skrive en meny som ser omtrent slik ut:

```
Meny
1: Registrer deltaker
2: Registrere tur for deltaker
9: Avslutt
Valg?
```

Når et valg er utført, skal menyen komme opp på nytt helt til brukeren velger å avslutte. I et reelt system, ville vi selvsagt hatt flere valg, men for å unngå for mye arbeid skal du ikke lage mer enn de tre menyvalgene som er skissert. Programmet skal skrives slik at det vil være lett å legge til flere valg. For å hjelpe dere litt i gang, viser vi hvordan en `HashMap` der vi brukar navn på deltakeren som nøkkel, kan deklareres og opprettes (instansieres).

```
HashMap<String, Deltaker> deltakere = new HashMap<String, Deltaker>();
```

For å få full score skal programmet lese inn nødvendige verdier og få de korrekt registrert (i HashMap-en), men husk at dere også kan få poeng ved for eksempel å få til deler av programmet. Når du skal lese inn verdier, kan du velge om du vil bruke Scanner-klassen eller metoder frå pakken easyIO.

Lykke til!

Vedlegg A, Deler av API-dokumentasjon for Dato

Class Dato

```
...
public class Dato
extends java.lang.Object
```

Enkel klasse for å behandle datoar.

Constructor Summary

Constructor and Description
`Dato(int dag, int mnd, int aar)` Opprettar ein dato der dag, mnd og år er gitt som parametrar.

Method Summary

Modifier and Type	Method and Description
`boolean`	`paaEllerEtter(Dato d)` Returnerer true dersom dato er på eller etter dato gitt som parameter.
`boolean`	`paaEllerFoer(Dato d)` Returnerer true dersom dato er på eller før dato gitt som parameter.

Vedlegg B, Deler av API-dokumentasjon for Tur

Class Tur

...

```
public class Tur
extends java.lang.Object
```

Tur er ein klasse for å registrere dato og postkode for ein tur.

Constructor Summary

Constructors

Constructor and Description
`Tur(Dato dato, int kode)` Opprettar ein tur der dato og kode er gitt som parametrar.

Method Summary

Methods

Modifier and Type	Method and Description
`Dato`	`getDato()` Returnerer ein dato for turen.
`int`	`getKode()` Returnerer postekode for turen.

Vedlegg C, Metoder for HashMap

`void`	**`clear`**`()` Removes all of the mappings from this map.
`Object`	**`clone`**`()` Returns a shallow copy of this `HashMap` instance: the keys and values themselves are not cloned.
`boolean`	**`containsKey`**`(`**`Object`**` key)` Returns `true` if this map contains a mapping for the specified key.
`boolean`	**`containsValue`**`(`**`Object`**` value)` Returns `true` if this map maps one or more keys to the specified value.
`Set<Map.Entry<K,V>>`	**`entrySet`**`()` Returns a **`Set`** view of the mappings contained in this map.
`V`	**`get`**`(`**`Object`**` key)` Returns the value to which the specified key is mapped, or `null` if this map contains no mapping for the key.
`boolean`	**`isEmpty`**`()` Returns `true` if this map contains no key-value mappings.
`Set<K>`	**`keySet`**`()` Returns a **`Set`** view of the keys contained in this map.
`V`	**`put`**`(`**`K`**` key, `**`V`**` value)` Associates the specified value with the specified key in this map.
`void`	**`putAll`**`(`**`Map`**`<? extends `**`K`**`,? extends `**`V`**`> m)` Copies all of the mappings from the specified map to this map.
`V`	**`remove`**`(`**`Object`**` key)` Removes the mapping for the specified key from this map if present.
`int`	**`size`**`()` Returns the number of key-value mappings in this map.
`Collection<V>`	**`values`**`()` Returns a **`Collection`** view of the values contained in this map.