Tentamen i Programmeringsteknik I, ES, 2010-03-18

- Skriv tid: 14-17.

- Hjälpmedel:
  1. Kurslitteratur, en av följande:
     - Lewis & Loftus, Java Software Solutions
     - Skansholm, Java Direkt
     - Guzdial & Ericson, Intrododuction to computing & programming.

  Litteraturen får inte innehålla överdrivet mycket anteckningar.

  2. Penna, radergummi, linjal.

- Tentamen består av fyra uppgifter och maxpoängen är 24.
- För att bli godkänd på tentamen (betyg 3) krävs minst 15 poäng
- För att få betyg 4 på tentamen krävs minst 18 poäng.
- För att få betyg 5 på tentamen krävs minst 21 poäng.

Instruktioner:
- Programmen skall vara skrivna i Java om inte annat anges.
- Programmen skall vara skrivna med vettig layout och förståeliga. Tydliggör gärna matchande måsvingar.
- Skriv varje uppgift på nytt papper.
- Skriv inte på baksidan.
- Använd inte rödpenna.
- Läs uppgifterna noga så du vet vad som skall göras.
- Gör inte mer än det som efterfrågas.
- Även ofullständiga lösningar och/eller lösningsskisser kan ge poäng om de visar att du tänkt rätt
- Läs igenom uppgiften igen när du löst den så att ser att du verkligen gjort rätt saker.
- Innan du lämnar in dina lösningar:
  - Lägg uppgifterna i rätt ordning.
  - Skriv namn på alla papper.
  - Skriv uppgiftens nummer i övre högra hörnet på varje blad.

Lycka till!

Torsten

Uppgift 1 (6p)

En stegräknare används för att bokföra antalet steg man tar vid en promenad. Antag att den innehåller följande tre egenskaper:

- Räknare för det totala antalet steg som man använt stegräknaren
- Räknare för antalet steg när stegräknaren senast användes.
- Namnet på den som äger stegräknaren

Skriv en klass `StegRaknare` som innehåller följande:

- Instansvariablerna `totRäknare`, `senastRäknare` och `ägare`.
- Två konstruktorer. En utan parameter och en med lämplig parameter (eller lämpliga parametrar).
- En metod `stega` som ökar båda räknarna med ett.
- En metod `nollställ` som nollställer instansvariabeln `senastRäknare`.
- En `toString`-metod för att erhålla information om stegräknaren.

Skriv också en mainmetod som

- Skapar en stegräknare med konstruktorn med parameter (parametrar).
- Stegräknaren skall sedan stegas 1000 steg.
- Skriv ut information om stegräknaren.
- Nollställ stegräknaren
- Skriv ut information om stegräknaren
- Stegräknaren stegas 500 steg
- Skriv ut information om stegräknaren.

Så här kan en körning av mainmetoden se ut:

```
Ägare=Kim.  Antal steg=1000. Totalt antal steg=1000
Ägare=Kim.  Antal steg=0. Totalt antal steg=1000
Ägare=Kim.  Antal steg=500. Totalt antal steg=1500
```

Uppgift 2 (6p)

a)
Givet är följande klass där vi enbart ser metodernas huvud:

```
public class EnKlass {

    ...

   public EnKlass() {
      ...
   }
   public void metodA(int h) {
      ...
   }
   public int metodB() {
      ...
   }
} // Slut på EnKlass
```

Här är en mainmetod som använder klassen:

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

   EnKlass e = new EnKlass();
   int[] x = {4, 5, 6};
   int tal = 2;

   e.metodA(tal);              // (1)
   e.metodA(x[3]);             // (2)
   e.metodA(x[tal]);           // (3)
   e.metodA(x);                // (4)
   int y = e.metodA(2) + 7;    // (5)
   int z = e.metodB() + 8;     // (6)
}
```

Vilka av satserna (1)-(6) är INTE tillåtna och varför?

b)
Givet är följande fungerande klass:

```
public class Vektor {

  private double x, y;

  public Vektor() {
    this.x = 0; this.y = 0;
  }

  public Vektor(double initX, double initY) {
    this.x=initX; this.y=initY;
  }

  public double getX() {
    return this.x;
  }

  public void setX(double newX) {
    this.x=newX;
  }

  public void add (Vektor other) {
     this.x = this.x + other.x;
     this.y = this.y + other.y;
  }

  public void scale (double factor) {
    this.x = this.x*factor;
    this.y = this.y*factor;
  }

  public String toString() {
    String str = "x="+this.x+" y="+this.y;
    return str;
  }

} // Slut klassen Vektor
```

Vad skrivs ut av följande fungerande mainmetod?

```
 public static void main (String[] arg ) {
    Vektor v1 = new Vektor();
    Vektor v2 = new Vektor(4.0,3.0);
    v2.setX( v1.getX()+1 );
    v2.scale(10.0);
    v2.add(v1);
    System.out.println(v1);
    System.out.println(v2);

 } // slut main
```

c)
En partikel kan med en enkel modell beskrivas som en cirkel med en centrumpunkt och en radie. En del av en klass Partikel kan då se ut så här:

```
public class Partikel {
   private double x,y,r;

   public Partikel() {
      this.x=0.0; this.y=0.0; this.r=1.0;
   }
   public Partikel(double inx, double iny, double inr) {
      this.x=inx; this.y=iny; this.r=inr;
   }
   ...
} // Slut klassen Partikel
```

En klass SpecialPartikel skall ärva klassen Partikel. Klassen SpecialPartikel skall ha en instansvariabel `vikt` som anger vikten på partikeln. Skriv klassen SpecialPartikel som enbart skall innehålla instansvariabeln och en konstruktor med parametrar för att bestämma centrumpunkt, radie och vikt.

Uppgift 3 (6p)
Givet är följande mainmetod:

```
public static void main (String[] arg ) {
    // Skapa en aktie med namnet Pollax med startkursen 29.0 kr (dag 0)
    Aktie a =  new Aktie("Pollax",29.0);
    a.nyKurs(26.8);           // Lägg in en ny kurs nästa dag (1)
    a.nyKurs(30.7);           // Lägg in en ny kurs nästa dag (2)
    a.nyKurs(29.3);           // Lägg in en ny kurs nästa dag (3)
    double max = a.max();     // Beräkna maximala värdet av kurserna
    System.out.println("Max aktiekurs: " + max);
    a.skrivUt();              // Skriver ut alla börskurser på skärmen
  }
```

En programkörning kan se ut så här:

```
Max aktiekurs: 30.7
Dag:0  Kurs:29.0
Dag:1  Kurs:26.8
Dag:2  Kurs:30.7
Dag:3  Kurs:28.8
```

Skriv klassen Aktie så att den passar mainmetoden. Klassen Aktie skall lagra alla kurser från dag 0 och framåt i en array. Antag att man aldrig någonsin är intresserad av att lagra fler än 10000 dagars kurser.

Uppgift 4 (6p)
Klassen Partikel finns beskriven i uppgift 2c.

a. Skriv en metod `kollision` i klassen Partikel som kontrollerar om två partiklar kolliderar eller inte. Metoden skall returnera `true` om partiklarna kolliderar annars skall metoden returnera `false`. Två partiklar anses kollidera med varandra om avståndet mellan partiklarnas centrumpunkter är mindre än summan av partiklarnas radier.
b. Skriv en `main`-metod som skapar 50 st Partikelobjekt med slumpvärden för position (x=0.0-100.0, y=0.0-100.0) och radie (1.0-5.0) samt lagra dem i en array. Därefter skall mainmetoden räkna och skriva ut hur många av dem som kolliderar med varandra. Metoden `kollision` skall användas för att testa ifall kollision föreligger.