UMEÅ UNIVERSITET
Inst. för Datavetenskap
Marie Nordström
Thomas Johansson

Juni 2002

Obligatorisk uppgift 1:
Grundläggande strukturer & språkelement

Redovisning

- Försättsblad med
  - Personnamn, användarnamn, kursens namn och uppgiftens nummer
- Innehållsförteckning
- Problemspecifikation
- Åtkomst och användarhandledning,
- Källkod
  - Prydligt indenterad och kommenterad (icke-proportionellt typsnitt, t.ex. `courier`, stl. 10) som i strukturen återspeglar enhetens arbete.
  - Filernas namn skall vara Pixel.java och Bild.java.
  - Metodernas namn och argument ska följa specifikationen.
  - Alla källkodsfiler ska finnas i katalogen *dinUser* / edu / javasu / lab1 /
- Testkörningar dokumenteras på lämpligt sätt, t.ex. med s.k. screen-shots (ctrl-Alt-PrintScreen)
- Systembeskrivning (t.ex. med UML-notation)
- Lösningens begränsningar
- Problem och reflektioner

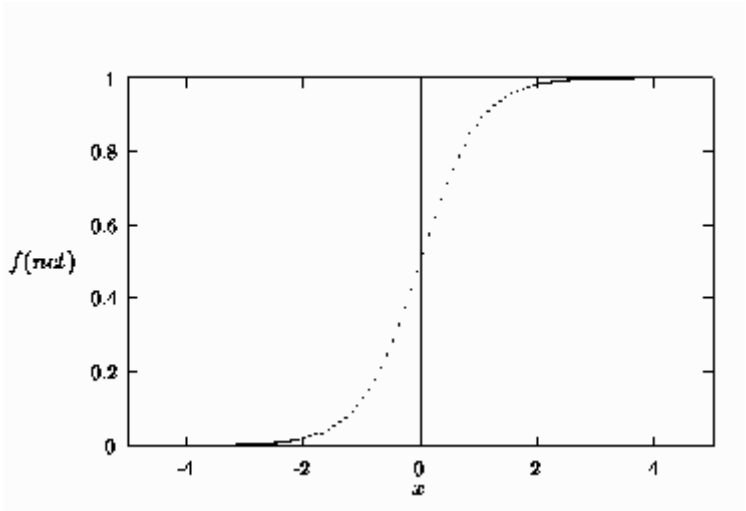
Uppgiften löses individuellt
och
inlämnas senast Torsdag 27/6 kl. 15.00

En digital bild är uppbyggd av bildpunkter, sk pixels, som kan ha olika färg. I vårt fall innehåller en pixel tre intensitetsvärden som representerar de tre grundfärgerna röd, grön, blå, och varje värde ligger i intervallet 0 till 63. 0 betyder inget bidrag från färgen, 63 betyder max färg, så en pixel med värden (röd=0, grön=0, blå=0) är helt svart och en pixel (röd=63, grön=63, blå=63) är helt vit. En röd pixel har värdena (röd=63, grön=0, blå=0) osv. Det finns alltså 64 * 64 * 64 olika färgkombinationer. Detta sätt att representera färger kallas RGB (red-green-blue)[1].

I denna uppgift skall du skriva en klass **`Pixel`** som kan representera en bildpunkt i en bild. Varje objekt av klassen Pixel skall ha information om styrkan på de tre grundfärgerna i just den punkten, och också metoder för att manipulera ljusstyrkan. Klassens konstruktor skall ta tre argument för värdet på intensiteten hos de tre färgerna.

Följande metoder skall finnas:

a) Skriv en metod **`kontrast`** som transformerar ett pixels tre färgintensiteter enligt följande sigmoidfunktion, som kan användas för att förändra kontrasten. En sigmoidfunktion ser ut som ett utdraget S:.



$$nyIntensitet = \frac{63}{1 + e^{-\beta \cdot (intensitet - \alpha)}}$$

där α och β är parametrar som bestämmer transformationen. α och β skall anges som argument till metoden, som skall ha följande signatur:

```
void kontrast(double alfa, double beta)
```

och kunna användas i följande programsegment:

```
Pixel pix = new Pixel(63, 20, 63);  // Lila ungefär
pix.kontrast(32, 2.5);
```

inspektera objektet pix efter anropet för att se om intensiteten ändrats.

[1] Om du är nyfiken på hur färger hanteras på skärmen kan du t.ex. titta på : http://www.w3.org/Graphics/Color/sRGB.html

b) Skriv en metod **`omkoda`** som klassificerar en pixel på så sätt att den bestämmer vilket intervall varje färg tillhör, och dess värde byts ut mot ett enda värde för varje intervall. Gränsvärdena finns i två fält som är max 63 element långa. Det ena fältet heter gränser och specificerar övre gräns för varje intervall, det andra fältet heter nykod och specificerar den nya intensiteten för alla värden i motsvarande intervall. Ett exempel på fält med längden 5 (båda fälten är alltid lika långa):

```
int[] gränser = { 20, 30, 40, 50, 63 };
int[] nykod = { 5, 10, 40, 45, 63 };
```

Detta innebär att en färg med intensitet i intervallet 0 – 20 byts ut mot värdet 5, en intensitet med värde 21 – 30 byts ut mot 10, osv. Du får 'bygga in' dessa fält i klassen Pixel. Metoden skall ha följande signatur:

```
void omkoda()
```

Metoden skall kunna användas i följande programsegment:

```
Pixel pix = new Pixel(60, 50, 20);  // Orange
pix.omkoda();
```

c) Skriv en metod **`svartvit`** som gör om en pixel från färg till svartvit, dvs sätter de tre intensitetsvärdena lika. Medelvärdet av intensiteten före och efter skall vara densamma (inom avrundningsfelens gränser, eftersom de är heltal).

d) Skriv en klass **`Bild`** som innehåller ett tvådimensionellt fält (32 x 32) med pixlar. Du kan låta Bild:s kontruktor skapa 1024 pixlar med slumpvis valda färger som du lägger i fältet.

e) Skriv en metod **`statistik`** till klassen Bild som på något sätt använder fältet `gränser` i deluppgift b för att redovisa statistik om sina pixlar. Varje gång metoden anropas skall den gå igenom alla pixlar i bilden och bestämma vilket intervall varje pixels röda värde tillhör, därefter skall intervallets ordningsnummer (dvs vilket index det blev i fältet gränser) användas för att räkna hur många pixlar i varje intervall det finns. Det kan hända att du måste komplettera klassen `Pixel` med fler metoder för att kunna lösa denna deluppgift. En metod **`antal`** används sedan för att fråga klassen om hur många pixlar den har i ett visst intervall. Metoderna skall ha följande signaturer:

```
void statistik()
int antal(int intervallNummer)
```

Ett exempel:

```
Bild bild = new Bild();  // Skapa Bild, med pixlar
bild.statistik();        // Beräkna statistik
bild.antal(0);           // Antal pixlar i intervall 0
```

Observera att alla attribut i både Bild och Pixel måste vara `private` !

Tips: delar av denna uppgift liknar delar av uppgift b, kan du återanvända något av koden?