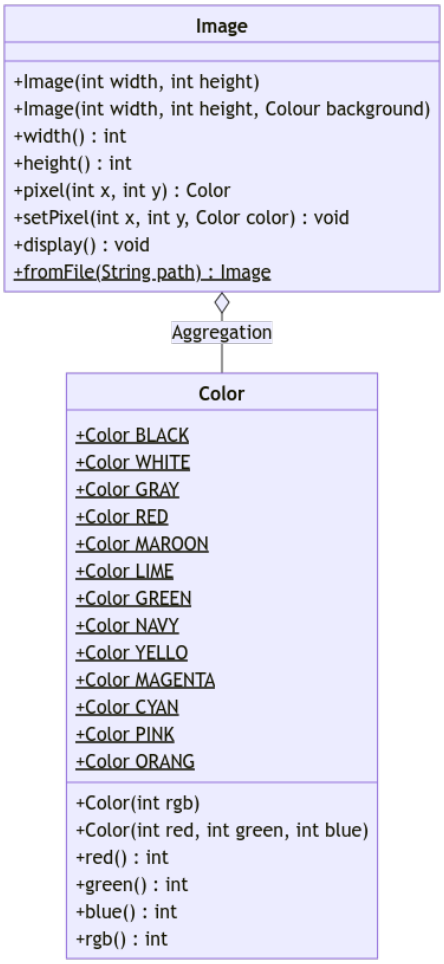
DM575: Øvelser

Øvelsessæt 3

Brug af klasser

En af de enkleste måder at repræsentere billeder på er som en todimensional række, der angiver for hver pixel på billedet dens røde, grønne og blå komponenter. Klassen Color giver faciliteter til at repræsentere farver ud fra netop deres røde, grønne og blå komponenter; den indeholder følgende medlemmer:

- en konstruktør med tre argumenter der opretter et objekt, der repræsenterer farven med de angivne røde, grønne og blå komponenter (forudsat at deres værdier er heltal i intervallet $[0, 255]$);
- en konstruktør med et argument, et heltal, der koder den røde, grønne og blå farve i dens tre mindst betydelige bytes (f.eks. `0xff0000` er rød, `0x00ff00` er grøn, og `0x0000ff` er blå
- metoder `int red()`, `int green()`, `int blue()` der returnerer værdien af den hhv. røde, grønne og blå komponent af farven;
- en metode `int rgb()` der returnerer et heltal der koder den røde, grønne og blå farve i dens tre mindst betydelige bytes;
- et antal klassevariable med instanser, der repræsenterer almindelige farver (gul, orange osv.).



Figur 1: UML klassediagrammer for `Image` og `Color`.

Bemærk, at den samme farve kan repræsenteres af forskellige instanser af `Color`; du bør derfor sammenligne farver ved hjælp af deres instansmetoden `equals` fremfor `==`. Klassen `Image` giver en simpel grænseflade til at manipulere billeder i dette format; den indeholder følgende medlemmer:

- en konstruktør med to argumenter, der opretter et sort billede med de angivne dimensioner;
- en konstruktør med tre argumenter, der opretter et billede med de angivne dimensioner og baggrundsfarve;
- en statisk metode med et argument, der tager stien til en fil, der indeholder et billede, og returnerer et objekt, der repræsenterer dette billede (eller `null`, hvis det ikke lykkes at indlæse filen);
- metoder `int width()` og `int height()` der returnerer bredden og højden af billedet;

- en metode `Color pixel(int x, int y)` der returnerer farven på den pixel med koordinaterne (x, y), under forudsætning af at disse koordinater repræsenterer en gyldig pixel;
- en metode `void setPixel(int x, int y, Color color)` der sætter farven på den pixel der har koordinaterne (x, y) til de angivne værdier (under forudsætning af at koordinaterne repræsenterer en gyldig pixel, og at farven ikke er `null`);
- en metode `void display()` der viser billedet på skærmen.

Vi anvender den konvention, at pixels angives fra øverste venstre hjørne af skærmen. Klasserne `Image` og `Color` indeholder *assertions* (`assert expr;`) til at kontrollere, at deres kontrakter overholdes. I Java er assertions som standard deaktiveret (i modsætning til Python) og kan aktiveres ved at kalde Java med flaget `-ea`, f.eks. `java -ea ImageUtils`.

1. Skriv en klasse `ImageUtils`, der implementerer nogle af de sædvanlige billedtransformationer, der er tilgængelige i de fleste billedredigeringsprogrammer. Implementer følgende (statistiske) metoder:

   (a) `Image flipHorizontal(Image image)` og `Image flipVertical(Image image)` der returnerer et nyt billede, hvor det oprindelige billede er vendt vandret eller lodret.

   (b) Tre rotationsmetoder `Image rotateLeft(Image image)`, `Image rotateHalf(Image image)` og `Image rotateRight(Image image)` der returnerer et nyt billede, der er opnået ved at rotere det oprindelige billede med hhv. en kvart omdrejning, en halv omdrejning eller en trekvart omdrejning mod uret.

   (c) En metode `Image stretchHorizontal(Image image)` der returnerer et nyt billede med samme højde som det oprindelige og dobbelt bredde, hvor hver pixel duplikeres horisontalt, og en tilsvarende metode `Image stretchVertical(Image image)` der gør det samme i vertikal retning.

   (d) En metode `Image crop(Image image, int x, int y, int width, int height)` der returnerer et nyt billede, der er opnået ved at beskære det oprindelige til det angivne område.

   (e) Tre metoder `Image switchRedGreen(Image image)`, `Image switchRedBlue(Image image)` og `Image switchGreenBlue(Image image)` der returnerer et nyt billede, hvor komponenterne for de to angivne farver er blevet byttet om.

   (f) Tre metoder `Image grayscaleAverage(Image image)`, `Image grayscaleLightness(Image image)`, og `Image grayscaleLuminosity(Image image)` der returnerer et nyt billede, hvor farverne er konverteret til gråtoner ved hjælp af hhv. gennemsnitsmetoden, lysstyrkemetoden og luminositetsmetoden:
      - gennemsnitsmetoden tildeler hver farvekomponent værdien $\frac{(r+g+b)}{3}$;
      - lysstyrkemetoden tildeler hver farvekomponent værdien $\frac{(\min(r,g,b)+\max(r,g,b))}{2}$;
      - luminositetsmetoden tildeler hver farvekomponent værdien $0{,}3 \cdot r + 0{,}59 \cdot g + 0{,}11 \cdot b$.

   (g) En metode `Color averageColor(Image image)` der beregner gennemsnitsfarven for det angivne billede, angivet ved den gennemsnitlige værdi af de røde, grønne og blå komponenter for hver pixel i billedet.

   (h) En metode `Image resample(Image image)` der returnerer et nyt billede, der er opnået fra det oprindelige som følger: for hver pixel der ikke er på kanten erstattes den røde komponent af gennemsnittet af de røde komponenter af de ni pixels i en 3×3 rektangel centreret omkring den aktuelle pixel. Den samme proces anvendes på de grønne og blå komponenter.

2. Skriv en hjælpeklasse `ImageUtilsSE` tilsvarende til den foregående, men hvor alle metoder returnerer `void` og ændrer det originale billede i stedet for at returnere et nyt. Bemærk at denne klasse kun inkluderer metoder der bevarer dimensionerne af billedet, da klassen `Image` ikke tillader disse at blive ændret efter instansiering. Implementer derudover følgende statiske metoder:

   (a) En metode `void addRectangle(Image image, int x, int y, int width, int height, Color color)` der tegner et rektangel i den angivne farve med størrelsen `width`×`height` således, at dens øverste venstre hjørne er placeret i positionen (x, y);

   (b) en metode `void addCircle(Image image, int x, int y, int radius, Color color)` der tegner en cirkel i den angivne farve af radius `radius` centreret om (x, y).

3. Skriv en hjælpeklasse `ImageCoder` med statiske metoder til at kryptere og dekryptere et billede ved hjælp af en given nøgle:

   (a) `void encrypt(Image image, String key)` der krypterer billedet som følger: udskift hver farvekomponent af hver pixel ved at lægge den sammen med samme farvekomponent af den tidligere besøgte pixel og til tegnkoden på det næste tegn i nøglen (modulo 256);

   (b) `void decrypt(Image image, String key)` der dekrypterer et billede under forudsætning af, at det blev krypteret af den forrige metode ved hjælp af den angivne nøgle.