Due: Tuesday, 9 April 2024, 11:59 PM

Naslov: Slike
Vrednost naloge: 20 točk

Slike in 256 odtenkov sive

Špela je bila danes spet zelo slabe volje - njen neotesani mlajši bratec je ponovno stikal po njenem računalniku in brskal po njenih slikah s poletnih počitnic na morju. To Špele niti ne bi toliko ujezilo, če na teh slikah ne bi bila tudi njena velika simpatija in le še tega se je manjkalo, da bo mali neotesanec bleknil kaj neprimernega v družbi njenih sošolk. Bratovo tokratno vedenje je bilo kaplja čez rob in Špela se je odločila temu narediti konec. Radovednemu bratu mora preprečiti vpogled v njene slike! Po nasvetu sosedovega Boštjana, ki obiskuje računalniško fakulteto in se spozna na te stvari, se je odločila, da bo vse slike zapisala v nestandardnem formatu p2, ki ga programi za pregledovanje slik ne poznajo, zato takih slik ne moremo kar tako preprosto odpreti in si jih ogledati.

Ker Špela ni preveč vešča programiranja, ji pomagajte napisati program za delo s slikami v formatu p2. Program naj omogoča branje slike, njen izpis, izračun nekaterih statističnih podatkov o sliki (histogram, povprečna svetlost) ter različne transformacije slike (zmanjšanje, zrcaljenje, rotiranje, zameglitev, iskanje robov).

Pomembno: Celotno nalogo implementirajte v eni datoteki `DN05.java`. Pred oddajo programa na eUčilnico preverite pravilnost delovanja. Da se testi izvedejo pravilno, bodite pozorni na pravilen zapis decimalnih vejic oziroma pik (kot je v primerih). Pri reševanju naloge se ne zanašajte le na pripravljene teste, saj vanje niso zajeti vsi možni primeri. Svojo rešitev tudi sami preverite, ali ustreza navodilom naloge!
Obvezno morate rešiti 1. nalogo v celoti, ostale naloge pa lahko rešujete v poljubnem vrstnem redu.

1. naloga (5 točk)

Za osnovno delo s slikami moramo znati prebrati podatke o sliki iz datoteke. Prebrane podatke shranimo v dvodimenzionalno tabelo in jih izpišemo na zaslon. Poleg tega izračunamo in izpišemo še histogram in svetlost slike.

Podrobnejša navodila naloge:

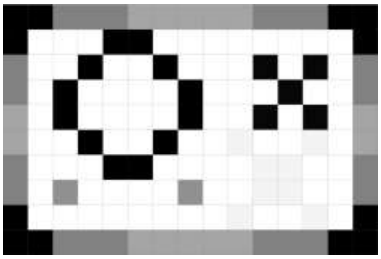
Format zapisa slike v datoteki

Sliko sestavlja zaporedje sivinskih pikslov, ki so razporejeni v mreži. Vsaka slika ima tudi določeno velikost, to je višino in širino. Širina pove, koliko pikslov imamo v eni vrstici (koliko je vseh stolpcev), višina pa pove, koliko pikslov imamo v enem stolpcu (koliko je vseh vrstic). Če je vrednost piksla 0, je to črn piksel, če je 255, pa bel piksel. Števila med 0 in 255 pa predstavljajo ustrezno sivino piksla.

Slika v formatu p2 je zapisana v tekstovni datoteki na naslednji način:

- v prvi vrstici je najprej zapisan format slike, to je niz "**P2**";
- za formatom je presledek in zapisana velikost slike v obliki *širina* x *višina*, npr. 800 x 600;
- *širina* in *višina* sta celi pozitivni števili (tip `int`);
- v drugi vrstici sledi zaporedje vrednosti posameznih pikslov (torej zaporedje *širina* x *višina* celih števil, ločenih s presledki); vsako celo število (tip `int`) predstavlja en piksel, ki ima lahko vrednosti med 0 in 255;
- piksli v sliki so zapisani od leve proti desni in od zgoraj navzdol.

Primer datoteke s sliko je *slika.p2*. Slika je velika 15 x 10 pikslov in je videti (zelo povečana) takole:



Metoda main()

Ustvarite razred `DN05` (v datoteki `DN05.java`) in v njem metodo `main()`, v kateri boste glede na podane argumente programa poklicali ustrezno metodo (podrobnosti so v nadaljevanju pri primeru klica za vsako nalogo).

Branje slike

Napišite metodo `int[][] preberiSliko(String ime)`, ki kot parameter prejme ime datoteke s sliko, zapisano v formatu p2. Metoda prebere podatke o pikslih in jih shrani v dvodimenzionalno tabelo, ki jo vrne kot rezultat.

Pri branju slike naj program izpiše napako, če slika ni v pravem formatu. Predvsem bodite pozorni na podpis slike (začne se z P2), napačno velikost (npr. 0, negativna ali prevelika - izven obsega tipa `int`), premalo podatkov o pikslih ... Če je v datoteki zapisanih več podatkov o pikslih, kot je velikost slike, preostale podatke zanemarite (in metoda še vedno vrne veljavno sliko, pri kateri upošteva le prvih *širina* x *višina* števil).

V primeru napak v podani datoteki *slika.p2*, naj program na standardni izhod izpiše naslednje napake:

- Če podane datoteke ne najde: "Napaka: datoteka slika.p2 ne obstaja."
- Če je podana datoteka prazna (obstaja, a nima vsebine): "Napaka: Datoteka slika.p2 je prazna."
- Če prva vrstica v datoteki ne ustreza opisanemu formatu P2: "Napaka: datoteka slika.p2 ni v formatu P2."
- Če velikost slike v prvi vrstici ne ustreza opisanemu formatu P2: "Napaka: datoteka slika.p2 ni v formatu P2 (velikost slike ni pravilna)."
- Če je podana širina ali višina slike 0 ali celo negativna: "Napaka: datoteka slika.p2 ni v formatu P2 (velikost slike je 0 ali negativna)."
- Če je podatkov o pikslih manj, kot je velikost slike: "Napaka: datoteka slika.p2 vsebuje premalo podatkov."
- Če je podatek o katerem koli pikslu napačen (negativen ali večji od 255): "Napaka: datoteka slika.p2 vsebuje podatke izven obsega 0 do 255."

Izpis slike

Napišite metodo `izpisiSliko(int[][] slika)`, ki kot parameter prejme sliko (dvodimenzionalno tabelo) in jo izpiše na standardi izhod. Izpis naj bo formatiran tako, da so podatki o vrednostih posameznih pikslov ustrezno podpisani, kot prikazuje primer.

Primer izpisa slike v datoteki *slika.p2* je naslednji:

```
velikost slike: 15 x 10
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
  0 255 255 255   0   0 255 255 255 255 255 255 255 255   0
128 255 255   0 255 255   0 255 255 255  16 255  16 255 128
128 255   0 255 255 255 255   0 255 255 255  16 255 255 128
164 255   0 255 255 255 255   0 255 255  16 255  16 255 164
164 255 255   0 255 255   0 255 255 244 255 255 244 255 164
128 255 255 255   0   0 255 255 255 255 244 244 255 255 128
128 255 144 255 255 255 255 144 255 255 244 244 255 255 128
  0 255 255 255 255 255 255 255 255 244 255 255 244 255   0
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
```

Poganjanje programa

Program poženete s pomočjo metode `main()` razreda `DN05`, ki naj izvede ustrezen del programa glede na ukaz, ki je prvi argument programa. Če je kot prvi argument podan niz `izpisi`, program prebere sliko iz podane datoteke (ime datoteke je drugi argument programa) ter prebrane podatke shrani v tabelo, kot je opisano zgoraj. Nato naj izpiše podatke na standardni izhod (klic metode `izpisiSliko()`).

Oblika izpisa je prikazana v naslednjem primeru. Ob klicu:

```
java DN05 izpisi slika.p2
```

program izpiše:

```
velikost slike: 15 x 10
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
  0 255 255 255   0   0 255 255 255 255 255 255 255 255   0
128 255 255   0 255 255   0 255 255 255  16 255  16 255 128
128 255   0 255 255 255 255   0 255 255 255  16 255 255 128
164 255   0 255 255 255 255   0 255 255  16 255  16 255 164
164 255 255   0 255 255   0 255 255 244 255 255 244 255 164
128 255 255 255   0   0 255 255 255 255 244 244 255 255 128
128 255 144 255 255 255 255 144 255 255 244 244 255 255 128
  0 255 255 255 255 255 255 255 255 255 244 255 255 244 255   0
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
```

Histogram

Napišite metodo `izpisiHistogram(int[][] slika)`, ki izračuna in izpiše histogram sivinske slike. Histogram je graf ali tabela, ki prikazuje zastopanost posameznih vrednosti na sliki (tj. koliko pikslov ima posamezno vrednost od 0 do 255). Pri izpisu histograma upoštevajte le tiste vrednosti sivin, ki se pojavijo v sliki.

Metodo `main()` dopolnite tako, da bo izvedla še ukaz `histogram`, pri katerem za podano sliko (klic metode `preberiSliko()`) na standardni izhod izpiše njen histogram (klic metode `izpisiHistogram()`).

Primer za sliko *slika.p2*, v kateri imamo 24 črnih pikslov (vrednost 0), 77 belih pikslov (vrednost 255), 0 pikslov z vrednostjo 1 (skoraj črni), 5 pikslov z vrednostjo 16 (zelo temno sivi) itd. Ob klicu:

```
java DN05 histogram slika.p2
```

program izpiše:

```
sivina : # pojavitev
    0  :   24
   16  :   5
  128  :   20
  144  :   2
  164  :   14
  244  :   8
  255  :   77
```

Svetlost

Napišite metodo `double svetlostSlike(int[][] slika)`, ki izračuna in vrne povprečno vrednost vseh pikslov na sliki.

Metodo `main()` dopolnite z obravnavo ukaza `svetlost`, ki prebere podano sliko, izračuna njeno svetlost (s klicem metode `svetlostSlike()`) ter rezultat izpiše na standardni izhod na dve decimalki natančno (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 svetlost slika.p2
```

program izpiše:

```
Srednja vrednost sivine na sliki slika.p2 je: 178.74
```

2. naloga (4 točke)

Pri urejanju slik nam pride prav tudi nekaj funkcij za manipulacijo s slikami, kot so zmanjšanje slike, njena rotacija ali zrcaljenje. Špelo pogosto zanima tudi to, v kateri vrstici na sliki je največja razlika v svetlosti pikslov. Dopolnite program še s temi funkcijami: zmanjšanje slike (1 točka), rotiranje slike (1 točka), zrcaljenje slike (1 točka) in vrstica z največjo razliko v svetlosti (1 točka).

Podrobnejša navodila naloge:

Zmanjšanje slike

Sliko vedno zmanjšamo na četrtino. To naredimo tako, da razpolovimo njeno širino in tudi višino. Če je širina ali višina liha, novo vrednost zaokrožimo navzdol (npr. pri sliki velikosti 15 x 10 bi bila zmanjšana slika velikosti 7 x 5). Napišite metodo `int[][] zmanjsajSliko(int[][] slika)`, ki podano sliko zmanjša na četrtino velikosti in vrne zmanjšano sliko. Če ima podana slika, ki jo želimo zmanjšati, *širino* ali *višino* manjšo od 3, metoda slike ne zmanjša, ampak vrne nezmanjšano sliko.

Metodo `main()` dopolnite z obravnavo ukaza `zmanjsaj`, ki prebere podano sliko, jo zmanjša (s klicem metode `zmanjsajSliko()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 zmanjsaj slika.p2
```

program izpiše:

```
velikost slike: 7 x 5
 63 191  73 209 209 191 159
191 127 255 127 255 135 195
209 127 255 127 252 195 192
191 227 127 227 255 244 255
 63 191 200 209 206 191 156
```

Rotiranje slike

Sliko vedno rotiramo za 90 stopinj v smeri urnega kazalca. Napišite metodo `int[][] rotirajSliko(int[][] slika)`, ki podano sliko rotira in vrne rotirano sliko.

Metodo `main()` dopolnite z obravnavo ukaza `rotiraj`, ki prebere podano sliko, jo rotira (s klicem metode `rotirajSliko()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 rotiraj slika.p2
```

program izpiše:

```
velikost slike: 10 x 15
  0   0 128 128 164 164 128 128   0   0
  0 255 255 255 255 255 255 255 255   0
128 255 144 255 255   0   0 255 255 128
128 255 255 255   0 255 255   0 255 128
128 255 255   0 255 255 255 255   0 128
164 255 255   0 255 255 255 255   0 164
164 255 255 255   0 255 255   0 255 164
164 255 144 255 255   0   0 255 255 164
164 255 255 255 255 255 255 255 255 164
164 244 255 255 244 255 255 255 255 164
128 255 244 244 255  16 255  16 255 128
128 255 244 244 255 255  16 255 255 128
128 244 255 255 244  16 255  16 255 128
  0 255 255 255 255 255 255 255 255   0
  0   0 128 128 164 164 128 128   0   0
```

Zrcaljenje slike

Sliko vedno zrcalimo horizontalno (to pomeni, da zamenjamo levo in desno stran, kot bi sliko gledali v ogledalu). Napišite metodo `int[][] zrcaliSliko(int[][] slika)`, ki podano sliko zrcali horizontalno in vrne zrcaljeno sliko.

Metodo `main()` dopolnite z obravnavo ukaza `zrcali`, ki prebere podano sliko, jo zrcali (s klicem metode `zrcaliSliko()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 zrcali slika.p2
```

program izpiše:

```
velikost slike: 15 x 10
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
  0 255 255 255 255 255 255 255 255   0   0 255 255 255   0
128 255  16 255  16 255 255 255   0 255 255   0 255 255 128
128 255 255  16 255 255 255   0 255 255 255 255   0 255 128
164 255  16 255  16 255 255   0 255 255 255 255   0 255 164
164 255 244 255 255 244 255 255   0 255 255   0 255 255 164
128 255 255 244 244 255 255 255 255   0   0 255 255 255 128
128 255 255 244 244 255 255 144 255 255 255 255 144 255 128
  0 255 244 255 255 244 255 255 255 255 255 255 255 255   0
  0   0 128 128 128 164 164 164 164 164 128 128 128   0   0
```

Vrstica z največjo razliko v svetlosti

V sliki poiščite tisto vrstico, kjer je razlika med najsvetlejšim in najtemnejšim pikslom (to imenujemo razlika v svetlosti) največja. Napišite metodo `int poisciMaxVrstico(int[][] slika)`, ki za podano sliko vrne indeks vrstice, v kateri je razlika v svetlosti pikslov največja. Če je takih vrstic več, metoda vrne najmanjši indeks.

V primeru slike *slika.p2* metoda `poisciMaxVrstico()` vrne 1. To pomeni, da ima druga vrstica (z indeksom 1) z največjo razliko v svetlosti. V tej vrstici so le črni in beli piksli (`0 255 255 255 255 255 255 255 255 0 0 255 255 255 0`), zato je razlika med najsvetlejšim (ki ima vrednost 255) in najtemnejšim (ki ima vrednost 0) enaka 255. To je tudi največja možna razlika, saj imamo na sliki le 256 sivin. Enako razliko v svetlosti dobimo tudi v tretji vrstici (in še več drugih vrsticah), a ima druga vrstica najmanjši indeks. V prvi vrstici z indeksom 0 (`0 0 128 128 128 164 164 164 164 164 128 128 128 0 0`) pa je najsvetlejši piksel 164 in najtemnejši 0, zato je razlika le 164 (in je manjša od razlike v svetlosti v drugi vrstici).

Metodo `main()` dopolnite z obravnavo ukaza `vrstica`, ki prebere podano sliko, v njej poišče vrstico z največjo razliko v svetlosti pikslov (s klicem metode `poisciMaxVrstico()`) in rezultat izpiše na standardni izhod (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 vrstica slika.p2
```

program izpiše:

```
Max razlika svetlo - temno je v 2. vrstici.
```

3. naloga (4 točke)

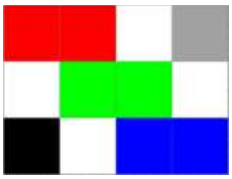
Špela bi želela imeti tudi barvne slike. Te so v formatu p2 zapisane podobno kot sivinske slike, le da je:

- v prvi vrstici zapisan format slike kot niz "**P2B**";
- vsak piksel predstavljen s celim številom (tip `int`), ki ima lahko vrednosti med 0 in 67.108.863;

Piksel v barvni sliki je predstavljen s trojico barvnih komponent: rdeče (R), zelene (G) in modre (B). Tako lahko en piksel predstavimo s trojico števil (R, G, B), kjer R pomeni količino rdeče barve (*red*), G količino zelene barve (*green*) ter B količino modre barve (*blue*). Vsaka barvna komponenta ima lahko vrednosti od 0 do 1023, torej skupaj 1024 različnih vrednosti. Tako za zapis vsake komponente potrebujemo 10 bitov, za vse tri komponente pa 30 bitov. Vse tri komponente so shranjene v podatkovnem tipu `int`, kjer sta prva dva bita vedno enaka 0, sledi 10 bitov komponente R, nato 10 bitov komponente G in na koncu še 10 bitov komponente B (gledano od leve proti desni, torej od najpomembnejšega bita proti najmanj pomembnemu):

```
   2 bita  10 bitov za R        10 bitov za G        10 bitov za B
     0 0  x x x x x x x x x x  x x x x x x x x x x  x x x x x x x x x x
```

Primer datoteke z barvno sliko je *slikaB.p2*. Slika je velika 4 x 3 piksle in je videti (zelo povečana) takole:



Dopolnite program še s funkcijami za branje in izpis barvnih slik (2 točki). Program naj omogoča tudi pretvorbo barvnih slik v sivinske (2 točki).

Podrobnejša navodila naloge:

Branje barvne slike

Napišite metodo `int[][][] preberiBarvnoSliko(String ime)`, ki kot parameter prejme ime datoteke s sliko, zapisano v formatu p2B. Metoda prebere podatke o pikslih in jih shrani v dvodimenzionalno tabelo. Vsak piksel je predstavljen s tabelo treh celoštevilskih vrednosti - komponent R, G in B. Torej je barvna slika pravzaprav trodimenzionalna tabela števil, ki jo metoda vrne kot rezultat.

Tudi tu pri branju slike preverite, ali je vhodna datoteka v pravem formatu. Predvsem bodite pozorni na podpis slike (začne se s P2B), napačno velikost slike, podane podatke o pikslih ... Program naj izpiše enake napake kot pri napakah sivinskih slik.

Izpis barvne slike

Napišite metodo `izpisiBarvnoSliko(int[][][] slika)`, ki kot parameter prejme barvno sliko (trodimenzionalno tabelo) in jo izpiše na standardni izhod. Izpis naj bo formatiran tako, da so podatki o vrednostih posameznih pikslov ustrezno podpisani (kot prikazuje primer izpisa slike *slikaB.p2* spodaj). Vsak piksel je izpisan kot trojica RGB komponent: (R, G, B).

Metodo `main()` dopolnite z obravnavo ukaza `barvna`, ki prebere barvno sliko iz podane datoteke (ime datoteke je drugi argument programa) in prebrane podatke shrani v tabelo, kot je opisano zgoraj. Nato naj izpiše podatke na standardni izhod (klic metode `izpisiBarvnoSliko()`).

Oblika izpisa je prikazana v naslednjem primeru. Ob klicu:

```
java DN05 barvna slikaB.p2
```

program izpiše:

```
velikost slike: 4 x 3
(1023,   0,   0) (1023,   0,   0) (1023,1023,1023) ( 641, 641, 641)
(1023,1023,1023) (   0,1023,   0) (   0,1023,   0) (1023,1023,1023)
(   0,   0,   0) (1023,1023,1023) (   0,   0,1023) (   0,   0,1023)
```

Pretvorba barvne slike v sivinsko

Barvno sliko lahko pretvorimo v sivinsko sliko tako, da za vsak piksel izračunamo povprečje vseh treh barvnih komponent in rezultat zaokrožimo navzdol na celo število. To je sivinska vrednost piksla. Ker pa ima v sivinski sliki piksel le 256 različnih vrednosti (vsaka RGB komponenta pa ima lahko 1024 različnih vrednosti), je potrebno izračunano vrednost še ustrezno preslikati na manjši interval zaloge vrednosti. Pri tem pazite, da bo bel piksel še vedno imel največjo vrednost (torej se barvni piksel (1023, 1023, 1023) preslika v sivinski piksel 255), črn piksel pa najmanjšo. Tudi pri tej preslikavi rezultat zaokrožite navzdol na celo število.

Število x z intervala [0, 1023] lahko pretvorimo v število y na intervalu [0, 255] z naslednjo enačbo:

$$y = x * (255 - 0)/(1023 - 0)$$

Napišite metodo `int[][] pretvoriVSivinsko(int[][][] slika)`, ki podano barvno sliko pretvori v sivinsko sliko, ki jo vrne kot rezultat.

Metodo `main()` dopolnite z obravnavo ukaza `sivinska`, ki prebere barvno sliko (argument programa), jo pretvori v sivinsko sliko (s klicem metode `pretvoriVSivinsko()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 sivinska slikaB.p2
```

program izpiše:

```
velikost slike: 4 x 3
 85  85 255 159
255  85  85 255
  0 255  85  85
```

4. naloga (2 točki)

Špela ima rada bolj temne slike, a kadar ima težek dan, si raje ogleduje svetle slike, saj jo vedno spravijo v dobro voljo. Zato bi želela svoje slike urediti tudi po svetlosti, da bo lažje izbirala slike glede na njeno razpoloženje. Programu dodajte še možnost, da vse podane slike uredi od najsvetlejše do najtemnejše. Svetlost slike določa vrednost povprečja vseh pikslov (glej ukaz `svetlost` pri 1. nalogi).

Podrobnejša navodila naloge:

Uredi podane slike

Napišite metodo `preberiVseInIzpisi(String[] imenaSlik)`, ki prebere slike iz datotek (metoda `preberiSliko()`), katerih imena so podana kot argumenti programa (prvi argument je ukaz, sledijo pa imena datotek, ki jih je lahko poljubno mnogo). Pri tem lahko predpostavite, da so vse podane datoteke v pravem formatu (in bo zato branje vseh slik uspešno). Metoda za vsako sliko izračuna njeno svetlost (klic metode `svetlostSlike()`) in rezultat zaokroži na celo število. Nato na standardni izhod izpiše imena slik, urejena od najsvetlejše do najtemnejše, ter zraven v oklepaju pripiše tudi svetlost slike. Če ima več slik enako svetlost, naj bodo pri izpisu urejene po abecedi (pri tem ne ločimo velikih in malih črk).

Metodo `main()` dopolnite z obravnavo ukaza `uredi`, ki pokliče metodo `preberiVseInIzpisi()`.

Primer. Ob klicu:

```
java DN05 uredi slika.p2 siva.p2 nova.p2 premajhna.p2 potovanje.p2 Prva.p2 mala.p2 crna.p2 bela.p2
```

program izpiše:

```
bela.p2 (255)
nova.p2 (255)
Prva.p2 (255)
slika.p2 (179)
mala.p2 (128)
potovanje.p2 (128)
premajhna.p2 (128)
crna.p2 (0)
siva.p2 (0)
```

5. naloga (5 točk)

Pogosto potrebuje Špela tudi bolj napredne načine (pre)urejanja slik, kot sta glajenje slike in poudarjeni robovi na sliki. Programu dodajte še te možnosti: izvedba osnovne konvolucije (1 točka), glajenje slike (2 točki) in iskanje robov (2 točki).

Podrobnejša navodila naloge:

Osnove konvolucije

Napišite metodo `void konvolucijaJedro(int[][] slika)`, ki bo izvedla operacijo konvolucije nad podano sliko. Konvolucija omogoča izvajanje različnih operacij nad sliko in se izvaja z uporabo jedra (angl. *kernel*), ki je dvodimenzionalna maska, pogosto velikosti 3x3, 5x5 ... Jedro potuje po sliki od zgornjega levega kota do spodnjega desnega kota, piksel po piksel, od leve proti desni in od zgoraj navzdol. Na vsakem koraku jedro pokrije sosednje piksle ter vsak piksel pomnoži z istoležnimi vrednostmi jedra in nato sešteje te zmnožke v eno število, ki predstavlja rezultat za trenutni sredinski piksel.

V metodi uporabite jedro velikosti 3x3, ki povsod vsebuje vrednost 1.

```
jedro: 3 x 3
1 1 1
1 1 1
1 1 1
```

Z jedrom se sprehodite preko slike po vsakem pikslu, od leve proti desni in od zgoraj navzdol ter shranjujte rezultate v novi tabeli. Pri tem pazite, da zaradi jedra velikost 3x3 ne padete izven slike, torej izračunajte rezultate le za piksle, ki so za en stolpec in eno vrstico odmaknjeni od roba. Na primer, če imamo vhodno sliko višine 4 in širine 4 piksle, izračunamo rezultate konvolucije le za

sredinske štiri piksle in dobimo rezultat višine 2 in širine 2 piksla. Rezultat konvolucije je torej velikosti (*širina* - 2) x (*višina* - 2). Rezultat prav tako ni sivinska slika, saj lahko vsebuje vrednosti, ki so večje od 255, vendar jo lahko še vedno obravnavamo kot sliko in jo izpišemo z metodo za izpis slike (`izpisiSliko()`). Pri izpisu rezultata lahko pride do težav, saj so lahko v rezultatu vrednosti s štirimi števkami in posledično stolpci v izpisu ne bodo pravilno podpisani, saj metoda za izpis izpisuje na 3 mesta. Tak izpis je pravilen in vam ga ni treba popravljati.

```
vhodna slika: 4 x 4
  0  64 128 255
255 128  64   0
  0   0   1   1
255 255  37   5
```

```
rezulat konvolucije: 2 x 2
640 641
995 491
```

Metodo `main()` dopolnite z obravnavo ukaza `jedro`, ki izračuna osnovno različico konvolucije nad sliko (s klicem metode `konvolucijaJedro()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 jedro slika.p2
```

program izpiše:

```
velikost slike: 13 x 8
1276 1531 1404 1185 1221 1512 1767 2022 1747 1711 1436 1547 1037
1531 1785 1530 1530 1530 1530 1785 2040 2056 1817 1578 1817 1547
1440 1530 1530 2040 2040 1530 1530 1785 1817 1578 1100 1578 1472
1476 1530 1530 2040 2040 1530 1530 1774 2045 1806 1567 1806 1736
1731 1785 1530 1530 1530 1530 1785 2029 2034 2023 1784 2034 1736
1839 1929 1674 1530 1530 1674 1929 2173 2262 2240 2240 2262 1939
1675 2184 1929 1785 1785 1929 2184 2173 2262 2240 2240 2262 1775
1165 1675 1803 1950 1986 1911 1911 1900 1964 1917 1881 1764 1265
```

Glajenje slike s konvolucijo

Napišite metodo `void konvolucijaGlajenje(int[][] slika)`, ki bo izvedla operacijo konvolucije za glajenje podane slike.

Za glajenje uporabite jedro velikosti 3x3 z naslednjimi vrednostmi:

```
jedro: 3 x 3
1/16  1/8 1/16
 1/8  1/4  1/8
1/16  1/8 1/16
```

Izračun konvolucije popravite, da bo zdaj vračal rezultat enake velikosti kot vhodna slika.

```
vhodna slika: 4 x 4
  0  64 128 255
255 128  64   0
  0   0   1   1
255 255  37   5
```

To dosežete tako, da vhodni sliki dodati po en stolpec na levi in desni strani ter po eno vrstico na vrhu in spodaj. V nova stolpca in novi vrstici se prepišejo vrednosti z robov vhodne slike in tako dobite razširjeno vhodno sliko velikosti (*širina* + 2) x (*višina* + 2):

```
razširjena slika: 6 x 6
  0   0  64 128 255 255
  0   0  64 128 255 255
255 255 128  64   0   0
  0   0   0   1   1   1
255 255 255  37   5   5
255 255 255  37   5   5
```

Rezultat glajenja s konvolucijo je zdaj enake velikosti kot vhodna slika:

```
rezultat konvolucije: 4 x 4
 68  84 124 172
116  88  68  64
120  86  37   7
192 151  63  10
```

Metodo `main()` dopolnite z obravnavo ukaza `glajenje`, ki izračuna konvolucijo nad sliko z uporabo jedra za glajenje (s klicem metode `konvolucijaGlajenje()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 glajenje slika.p2
```

program izpiše:

```
velikost slike: 15 x 10
 16  72 136 144 119 133 172 188 188 181 167 160 136  72  16
 72 160 200 160 114 119 169 217 233 216 196 194 186 145  72
136 200 192 160 176 176 160 192 240 226 181 166 181 186 136
167 178 144 176 240 240 176 144 208 226 166 136 166 196 167
181 183 144 176 240 240 176 144 207 225 180 165 180 200 181
181 215 192 160 176 176 160 192 239 237 223 223 222 215 181
167 219 226 185 144 144 185 226 248 253 249 249 253 225 167
136 202 228 226 208 208 226 228 241 253 249 249 253 215 136
 72 153 202 217 226 231 226 219 225 227 223 221 212 159  72
 16  72 136 160 167 181 188 188 187 180 166 159 135  71  16
```

Iskanje robov s konvolucijo

Napišite metodo `void konvolucijaRobovi(int[][] slika)`, ki bo izvedla operacijo konvolucije za iskanje robov v podani vhodni sliki.

```
vhodna slika: 4 x 4
  0  64 128 255
255 128  64   0
  0   0   1   1
255 255  37   5
```

Najprej bomo s konvolucijo iskali navpične robove z naslednjim jedrom:

```
jedroX: 3 x 3
 1  0 -1
 2  0 -2
 1  0 -1
```

Rezultat konvolucje oz. navpične robove shranimo v tabelo `roboviNavpično` (dobljene vrednosti niso več na intervalu med 0 in 255, kar je pravilno):

```
roboviNavpično: 4 x 4
-65 -193 -445 -317
190 253  64   1
127 407 376  96
  0 653 749  96
```

Nato poiščemo še vodoravne robove z naslednjim jedrom:

```
jedroY: 3 x 3
 1  2  1
 0  0  0
-1 -2 -1
```

Rezultat konvolucje oz. vodoravne robove shranimo v tabelo `roboviVodoravno` (dobljene vrednosti niso več na intervalu med 0 in 255, kar je pravilno):

```
roboviVodoravno: 4 x 4
-829 -319 319 829
 64 255 572 889
-127 -227 -78  12
-1020 -801 -331 -48
```

Rezultata združimo v novo tabelo `roboviSkupaj` enake velikosti tako, da vsak element tabele izračunamo kot kvadratni koren vsote kvadratov istoležnih rezultatov navpičnih robov in vodoravnih robov (glej spodnjo formulo). Preden rezultat shranimo v novo tabelo, ga še zaokrožimo na najbližje celo število.

$$tabela[i][j] = \sqrt{roboviNavpično[i][j]^2 + roboviVodoravno[i][j]^2}$$

```
roboviSkupaj: 4 x 4
832 373 548 888
200 359 576 889
180 466 384  97
1020 1033 819 107
```

Dobljene vrednosti so lahko večje od največje možne vrednosti sivinske slike, zato je potrebno vrednosti preslikati v vrednosti med 0 in 255. Minimalna meja trenutnega rezultata je 0, za maksimalno mejo pa vzamemo največjo vrednost v tabeli `roboviSkupaj`. V splošnem to pretvorbo števil naredimo z enačbo, kjer imamo število c, ki ima vrednosti med a in b, in ga pretvorimo v novo število x, ki ima vrednosti med y in z:

$$x = (c - a) * (z - y)/(b - a) + y$$

Rezultat je tabela `roboviKončni`, ki ima vse elemente med 0 in 255 (kar ustreza sivinski sliki):

```
roboviKončni: 4 x 4
205  92 135 219
 49  89 142 219
 44 115  95  24
252 255 202  26
```

Metodo `main()` dopolnite z obravnavo ukaza `robovi`, ki s pomočjo konvolucije v sliki poišče robove (s klicem metode `konvolucijaRobovi()`) in rezultat izpiše na standardni izhod (s klicem metode `izpisiSliko()`) (glej spodnji primer izpisa).

Primer. Ob klicu:

```
java DN05 robovi slika.p2
```

program izpiše:

```
velikost slike: 15 x 10
 85 213 175  85  77 121  65  86  86  98 114 120 175 213  85
213 255  95 120  73  77 125  65  86  75   8   7  48 175 213
175  95 170   0 190 190   0 170  85 113  80 113  80  48 175
114  86  85 190  85  85 190  85 190 113 113   0 113   8 114
 98 110  85 190  85  85 190  85 187 113  76 110  76  22  98
 98  62 170   0 190 190   0 170  81  80 105 105 110  58  98
114  92  61 133  85  85 133  61  33   8   7   7   8 109 114
175 102   0  61 190 190  61   0  50   8   7   7   8 151 175
213 218 102  97 111  95  65  33  63  92 104 112 150 251 213
 85 213 175 120 114  98  86  86  83  93 111 117 171 209  85
```

Datoteke, ki so uporabljene v pripravljenih testih, so zbrane v viri.zip.

Špela je tudi že napisala program, ki prikaže podano p2 sliko (ShowImage.jar). Programu lahko podamo (kot drugi argument) tudi želeno višino prikazane slike (privzeta vrednost je 800) in slika se ustrezno poveča oz. pomanjša. Morda vam ta program pride prav pri izdelavi domače naloge. Poženete ga takole:

```
java -jar ShowImage.jar ime_slike.p2 [višina_prikazane_slike]
```

Add submission

Submission status

Attempt number	This is attempt 1.
Submission status	No submissions have been made yet
Grading status	Not graded
Time remaining	7 days 5 hours remaining

◄ Rešitev: DN04.java

Jump to...

Zanke (in uganke) ►