9. homework assignment; JAVA, Academic year 2014/2015; FER

This homework has two problems.

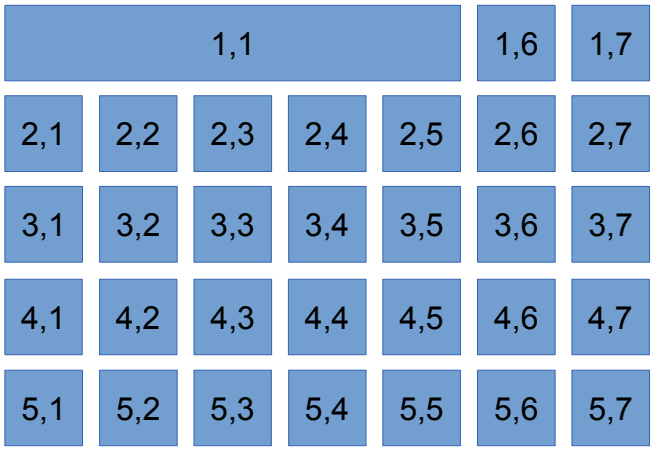
Zadatak 1.

Za potrebe rješavanja ovog zadatka (i sljedećeg koji rješavate u istom projektu) napravite novi Maven projekt: groupId **`hr.fer.zemris.java.studentVASJMBAG.hw09`**, artifactId **`calculator`**.

Proučiti:

http://docs.oracle.com/javase/tutorial/uiswing/layout/custom.html

Sve komponente razvijete u okviru ovog zadatka stavite u paket `hr.fer.zemris.java.gui.layouts`. Napravite vlastiti *layout manager* naziva `CalcLayout`, koji će implementirati sučelje `java.awt.LayoutManager2`. Ograničenja s kojima on radi trebaju biti primjerci razreda `RCPosition` koji nudi dva *read-only* svojstva: `row` te `column` (oba po tipu `int`). Za raspoređivanje komponenti layout manager konceptualno radi s pravilnom mrežom dimenzija 5 redaka i 7 stupaca (ovo je fiksirano i nije moguće mijenjati). Numeracija redaka i stupaca kreće od 1. Izgled layouta je prikazan na slici u nastavku, a u komponentama su upisane njihove koordinate.



Svi retci mreže su jednako visoki; svi stupci mreže su jednako široki. Podržano je razmještanje najviše 31 komponente. Pri tome se komponenta koja je dodana na poziciju (1,1) uvijek razmješta tako da prekriva i pozicije (1,2) do (1,5); to znači da pokušaj dodavanje komponenti uz ograničenja (1,2) do (1,5) treba izazvati odgovarajuću iznimku (kao i bilo koja druga nelegalna pozicija, tipa (-2,0) ili (1,8) ili ...).

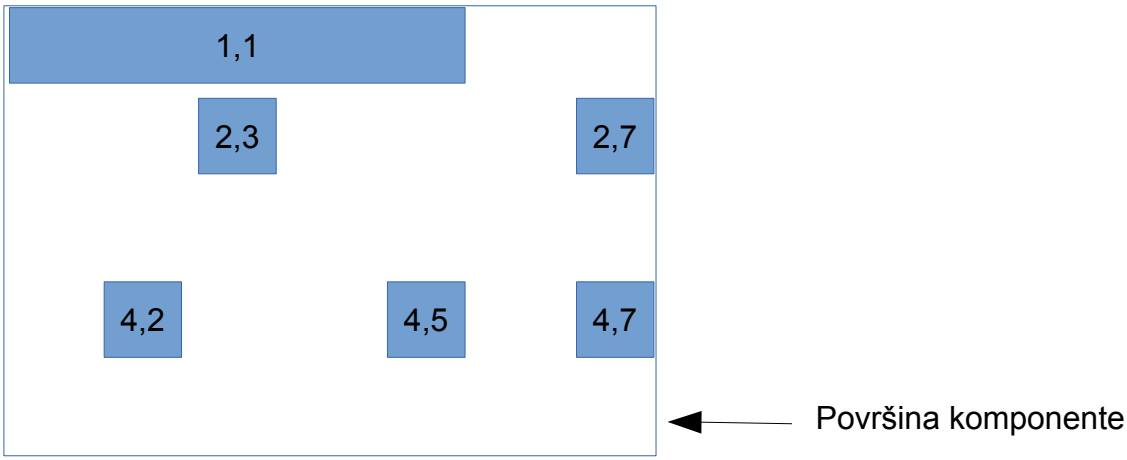
Pri izračunu preferiranih dimenzija layouta držite se sljedećih pretpostavki. Visina svih redaka je ista i određuje se kao maksimalna visina od preferiranih visina svih dodanih komponenti. Širina svih stupaca je ista i određuje se kao maksimalna širina od preferiranih širina svih komponenata (izuzev one na poziciji (1,1) – pazite kako taj broj morate interpretirati). Pri stvarnom raspoređivanju, moguće je da komponenta neće biti baš preferirane veličine: u tom slučaju proporcionalno skalirajte veličine tako da komponente popune kontejner u koji su dodane.

Prilikom raspoređivanja, legalno je da nisu prisutne sve komponente layouta; dapače, legalno je i da su čitavi retci ili stupci prazni – to ništa ne mijenja (i u layoutu na tim mjestima stvarno ostaju prazna polja).

Primjerice, sljedeći kod:

```
JPanel p = new JPanel(new CalcLayout(3));
p.add(new JLabel("x"), new RCPosition(1,1));
p.add(new JLabel("y"), new RCPosition(2,3));
p.add(new JLabel("z"), new RCPosition(2,7));
p.add(new JLabel("w"), new RCPosition(4,2));
p.add(new JLabel("a"), new RCPosition(4,5));
p.add(new JLabel("b"), new RCPosition(4,7));
```

bi trebao generirati razmještaj prikazan u nastavku.



`CalcLayout` mora podržati dva konstruktora: jedan koji prima željeni razmak između redaka i stupaca (u pikselima; tip `int`), tako da se može postići razmještaj u kojem komponente nisu zalijepljene jedna za drugu – on je prikazan u prethodnom primjeru. Drugi bez argumenata koji ovaj razmak postavlja na 0.

Prilikom stvaranja komponente koja koristi ovaj layout nužno je najprije nad komponentom instalirati ovaj layout manager (bilo kroz konstruktor ako to komponenta podržava, ili pozivom `.setLayout(...)`), i tek potom krenuti u dodavanje komponenti.

Potrebno je podržati i ograničenja koja se zadaju u obliku stringa koji tada mora biti propisane strukture. Primjerice, sljedeći kod bi morao stvoriti identičan layout prethodno prikazanome.

```
JPanel p = new JPanel(new CalcLayout(3));
p.add(new JLabel("x"), "1,1");
p.add(new JLabel("y"), "2,3");
p.add(new JLabel("z"), "2,7");
p.add(new JLabel("w"), "4,2");
p.add(new JLabel("a"), "4,5");
p.add(new JLabel("b"), "4,7");
```

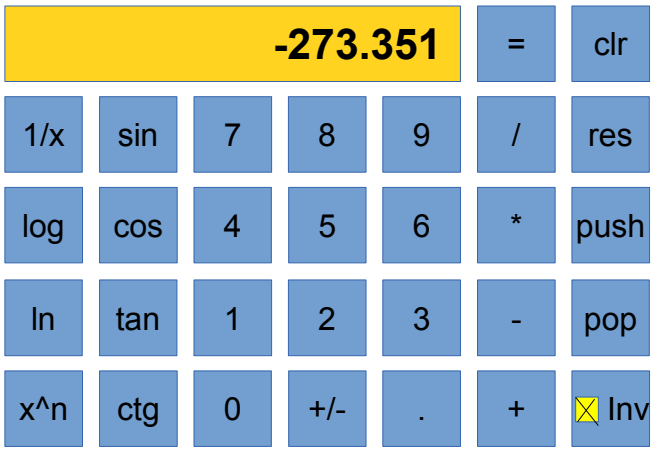
Ako korisnik pokuša dodati dvije komponente pod istim ograničenjem, layout manager bi trebao izazvati iznimku. Ako se layout manager instalira u kontejner koji već sadrži druge komponente (za koje naš layout manager ne zna), slobodan ih je ignorirati.

Metode layout managera kojima bi on trebao kontejneru vratiti informacije o preferiranoj veličini layouta, te minimalnoj i maksimalnoj potrebno je izvesti malo pažljivije (za početak, obratite pažnju da neka komponenta kada ju pitate za neku od tih informacija može vratiti i `null`, čime poručuje da joj to nije bitno, odnosno nema nikakvog zahtjeva). Minimalna veličina layouta mora garantirati svakoj komponenti koja se izjasni da ima barem toliko mjesta – razmislite što to znači. Analogno vrijedi i za maksimalnu veličinu.

Zadatak 2.

Rješavanje nastavljate u istom projektu u kojem ste riješili prethodni zadatak.

Uporabom prethodno razvijenog layout managera napišite program `Calculator` (razred `hr.fer.zemris.java.gui.calc.Calculator`). Budete li stvarali dodatne razrede, stavite ih u isti paket u kojem je ovaj razred ili u njegove potpakete. Skica prozora kalkulatora prikazana je u nastavku.



Kalkulator funkcionira kao onaj standardni Windows kalkulator: broj se unosi klikanjem po gumbima sa znamenkama. "Ekran" koji prikazuje trenutni broj je komponenta `JLabel`: nije omogućen unos broja utipkavanjem preko tipkovnice. Primjerice, da biste izmnožili 32*2, naklikat ćete 3, 2, puta, 2, = i dobiti 64. Ako naklikate 3, 2, *, 2, + 1, =, na ekranu će se prikazati redom, "3", "32", "32" (sustav pamti operaciju *), "2", "64" (i sustav pamti operaciju +), "1", "65".

Tipka "clr" briše samo trenutni broj (ali ne poništava operaciju; primjerice, 3, 2, *, 5, 1, clr, 2, = će i dalje izračunati 64; krenuli smo množiti s 51, predomislili se i pomnožili s 2). Tipka "res" kalkulator resetira u početno stanje. Tipka push trenutno prikazani broj stavlja na stog; npr. 3, 2, * 2, push, =, na stog stavlja 2, na jednako ispisuje 64. Tipka pop trenutni broj mijenja brojem s vrha stoga (ili dojavljuje da je stog prazan). Checkbox Inv (komponenta **`JCheckBox`**) obrće značenje operacija *sin*, *cos*, *tan*, *ctg* (u arkus funkcije), *log* u 10^, *ln* u *e*^, *x*^*n* u *n*-ti korijen iz *x*. Trigonometrijske funkcije podrazumijevaju da su kutevi u radijanima. Za pohranu brojeva koristite tip *double* (ne trebate implementirati podršku za velike brojeve; nije poanta ovog zadatka).

Uočite da dosta prikazanih tipki konceptualno radi vrlo slične obrade (tipa: pritisak na znamenku; pritisak na neku od tipki koje odmah djeluju poput "sin" koji odmah računa sinus trenutno prikazanog broja i na ekran zapisuje rezultat, i slično). Identificirajte takve grupe sličnih operacija i napišite generički kod u kojem ćete konkretno djelovanje (tipa: računam li sinus ili kosinus ili njima inverzne operacije) modelirati odgovarajućom *strategijom* (oblikovni obrazac Strategija).

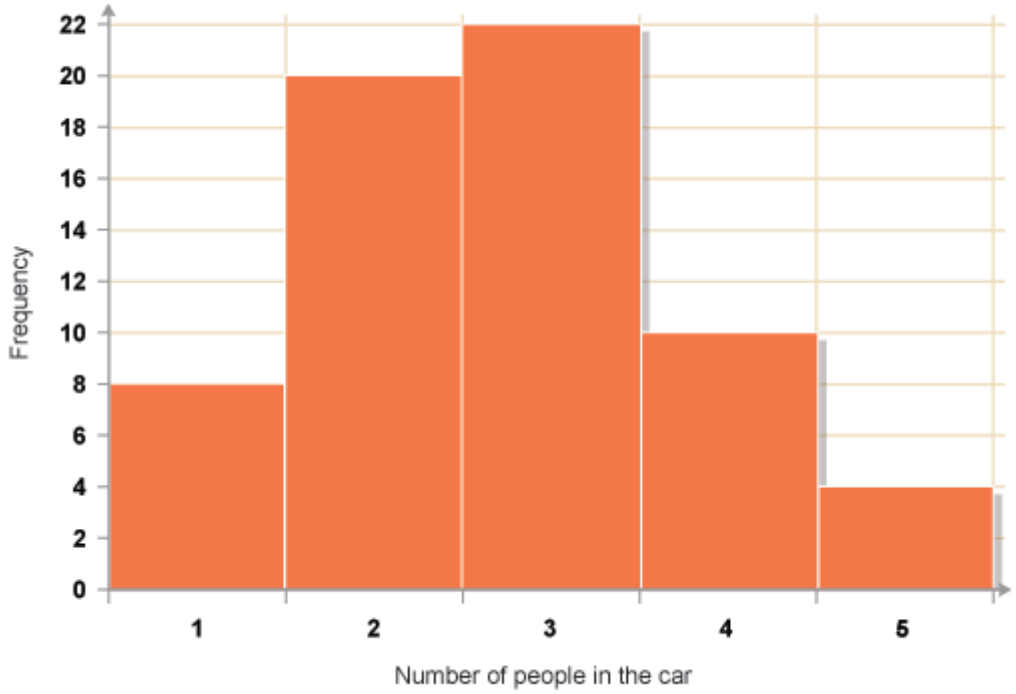
Zadatak 3.

Za potrebe rješavanja ovog zadatka napravite novi Maven projekt: groupId **`hr.fer.zemris.java.studentVASJMBAG.hw09`**, artifactId **`barchart`**.

U paketu `hr.fer.zemris.java.gui.charts` definirajte razred `XYValue` koji ima dva read-only property-ja: `x` i `y`, oba tipa `int`. Definirajte potom u istom paketu razred `BarChart` koji kroz konstruktor prima listu `XYValue` objekata, opis uz x-os te opis uz y-os, minimalni y koji se prikazuje na osi, maksimalni y koji se prikazuje na osi te razmak između dva susjedna y-a koji se prikazuju na osi (ako ymax-ymin ne dijeli taj razmak, pri crtanju radite s prvim većim y-onom koji je na cijelom broju razmaka od ymin).

Primjerice, podatke za dijagram prikazan na sljedećoj slici konstruirali bismo pozivom:

```
BarChart model = new BarChart(
  Arrays.asList(
    new XYValue(1,8), new XYValue(2,20), new XYValue(3,22),
    new XYValue(4,10), new XYValue(5,4)
  ),
  "Number of people in the car",
  "Frequency",
  0,      // y-os kreće od 0
  22,     // y-os ide do 22
  2
);
```



U paketu `hr.fer.zemris.java.gui.charts` definirajte razred `BarChartComponent` koji je izveden iz razreda JComponent. Ima jedan konstruktor koji prima referencu na BarChart i pamti je. Komponenta na svojoj površini stvara prikaz podataka koji su definirani primljenim objektom (tj. crta stupičasti dijagram). Pri izradi crteža, vodite se sljedećim zahtjevima.

- Komponenta s lijeve strane ispisuje naziv podataka s y-osi, slijedi neki (fiksni) razmak, vrijednosti po y-osi koje su desno poravnate (kao na slici), neki fiksni razmak, pa y-os.

- Komponenta s donje strane (gledano od samog dna) ispisuje naziv podataka s x-osi, slijedi neki (fiksni) razmak, vrijednosti po x-osi koje su horizontalno centrirane ispod stupića (kao na slici), neki fiksni razmak, pa x-os.
- Za dio osi x i y koji su na kraju (gdje je strelica) uzmite također neki fiksni iznos (ili ga prilagodite po potrebi; tu imate slobodu).

Posljedica ova tri zahtjeva je sljedeća: kada se komponenta razvlači, udaljenost osi y od lijevog ruba te osi x od dna ostaje fiksna: mijenja se samo visina y-osi, širina x-osi te sam prostor u kojem se prikazuju stupići (čime i stupići postaju širi/uži, viši/niži). Slika uvijek mora popunjavati cjelokupnu površinu komponente. Pri razvlačenju komponente nemojte ništa mijenjati oko fontova: font (i veličina fonta) neka uvijek bude ista.

Ne smijete unaprijed ništa pretpostavljati o broju znamenaka vrijednosti koje se ispisuju na y-osi (osim da je sve lijepo prikazivo int-om).Udaljenost osi y od lijevog ruba mora "plesati" ovisno o stvarno potrebnom prostoru za prikaz brojeva. Rješenje koje unaprijed alocira fiksan broj (npr. 300 piksela, za svaki slučaj) u nadi da će svi brojevi biti prikazivi nije prihvatljivo.

U paketu `hr.fer.zemris.java.gui.charts` definirajte razred `BarChartDemo` koji je izveden iz `JFrame` i prikazuje preko čitave svoje površine jedan primjerak grafa. Dodajte tom razredu i metodu `main` tako da sve možete pokrenuti iz naredbenog retka. Program prima jedan argument: stazu do datoteke u kojoj je opis grafa. Primjer takve datoteke (za prethodno prikazani graf) prikazan je u nastavku (sve između vodoravnih crta). Program otvara datoteku, temeljem sadržaja stvara objekt `BarChart`, njega predaje prozoru koji dalje radi što treba. Uz gornji vrh prozora stavite jednu labelu u kojoj će biti ispisana (vodoravno centrirano) putanja do datoteke iz koje su podatci dohvaćeni.

```
Number of people in the car
Frequency
1,8 2,20 3,22 4,10 5,4
0
22
2
```

U datoteci gledate prvih 5 redaka. Točke su međusobno odvojene razmacima a komponente točke zarezom. Ako bilo što ne štima s formatom datoteke koju program dobije, javite to korisniku i prekinite izvođenje.

Pomoć: vertikalni ispis teksta:
http://www.codejava.net/java-se/graphics/how-to-draw-text-vertically-with-graphics2d
(nemojte nakon aktiviranja rotacije i ispisa teksta zaboraviti poništiti daljnje rotiranje).

PS. Ne morate bacati sjenu (kao na slici) ako nemate ideju kako to napraviti.

Zadatak 4.

Za potrebe rješavanja ovog zadatka napravite novi Maven projekt: groupId **`hr.fer.zemris.java.studentVASJMBAG.hw09`**, artifactId **`prim`**.

Razrede smještate u paket `hr.fer.zemris.java.gui.prim`.

Napišite program `PrimDemo`: to je program koji se pokreće iz naredbenog retka bez ikakvih argumenata. Po pokretanju otvara prozor u kojem se prikazuju dvije liste (jednako visoke, jednako široke, rastegnute preko čitave površine prozora izuzev donjeg ruba). Uz donji rub dodajte gumb "sljedeći".

Napišite model liste `PrimListModel` koji nudi metodu `next()`. Model inkrementalno generira prim brojeve (po stvaranju, u modelu se nalazi samo broj 1). Na svaki poziv metode `next()` model u popis brojeva doda prvi sljedeći prim broj. Prozor koji prikazuje liste treba obje liste registrirati nad istim primjerkom modela (drugim riječima, smijete stvoriti samo jedan primjerak razreda `PrimListModel`. Klik na gumb poziva metodu `next()` modela; kao posljedica, obje liste moraju prikazati i taj novododani broj. Skrolovi se trebaju pojaviti automatski kada su potrebni. Razred `PrimListModel` morate izvesti iz vršnog sučelja modela i sami morate implementirati svu potrebnu funkcionalnost (nije dozvoljena uporaba razreda koji već nude dio funkcionalnosti).

Important notes

You must create a single ZIP archive containing all projects which you have created as part of this homework (each in its own folder), and than upload this single ZIP. ZIP arhive must have name `HW09-`*yourJMBAG*`.zip`.

All of the classes should have appropriate javadoc. No test are required (but are strongly advised, where appropriate).

Please note. You can consult with your peers and exchange ideas about this homework *before* you start actual coding. Once you open you IDE and start coding, consultations with others (except with me) will be regarded as cheating. You can not use any of preexisting code or libraries which is not part of Java standard edition (Java SE) unless explicitly allowed or provided by me. You can use Java Collection Framework classes and its derivatives. Document your code!

Upload final ZIP archive to Ferko before the deadline. Do not forget to lock your upload or upload will not be accepted. Deadline is May 16nd 2015. at 07:00 AM.