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 ...).

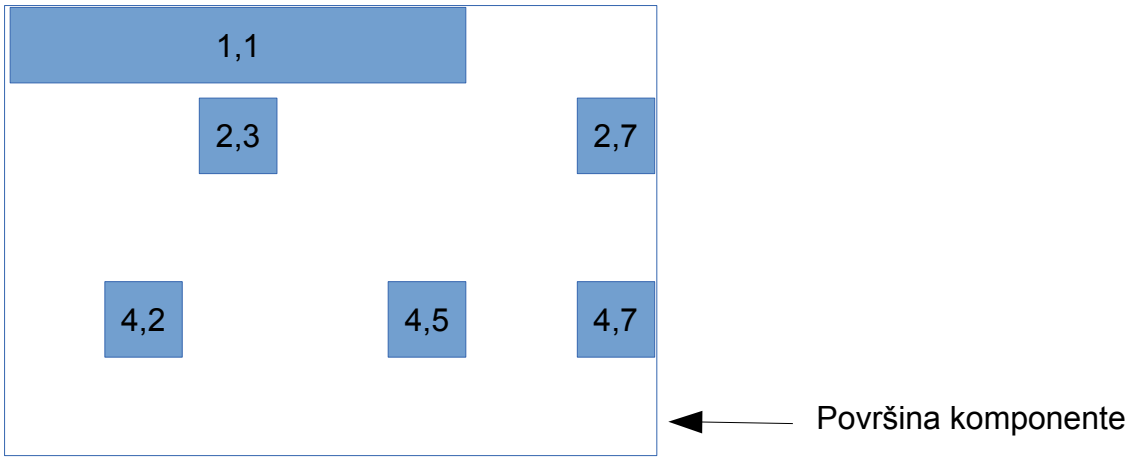
Pri izračunu preferiranih dimenzija layouta držite se sljedećih pretpostavki. Visina svih redaka je ista i određuje se kao maksimalna visina od preferiranih visina svih dodanih komponenti. Širina svih stupaca je ista i određuje se kao maksimalna širina od preferiranih širina svih komponenata (izuzev 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");
```

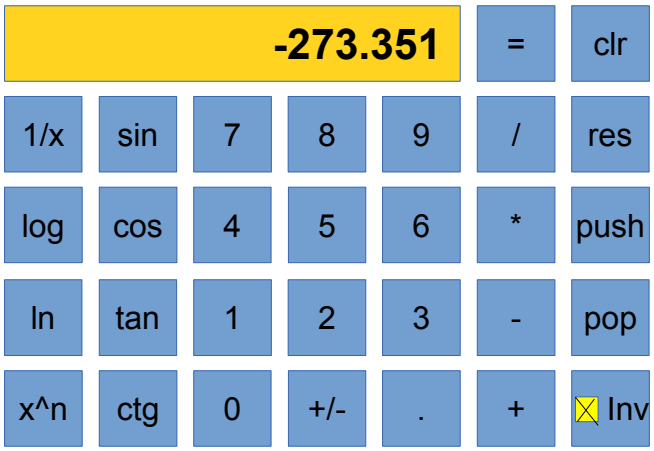
Ako korisnik pokuša dodati dvije komponente pod istim ograničenjem, layout manager bi trebao izazvati iznimku. Ako se layout manager instalira u kontejner koji već sadrži druge komponente (za koje naš layout manager ne zna), slobodan ih je ignorirati.

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).