Završni ispit

Sve zadatke rješavate u Vašoj 8. domaćoj zadaći. Stoga napravite sljedeću pripremnu radnju.

1. U 8. domaćoj zadaći napravite novi paket `ispit`.
2. U taj paket dodajte glavni program – razred `Pokretac`.
3. U njega dodajte metodu `main`, te osigurajte da se izvođenjem te metode pokrene Vaš notepad. Ako ste prije toga za to imali u paketu `x` razred `Y`, najjednostavnije je da iz metode main razreda `Pokretac` naprosto pozovete metodu main razreda `x.Y`.

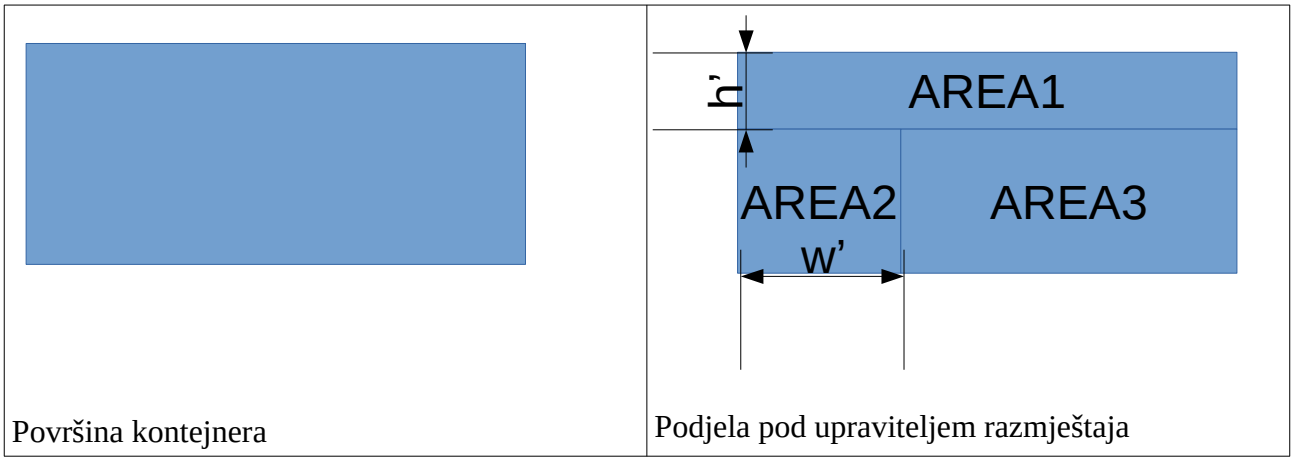
Ovime će se osigurati da se čitavo rješenje uvijek može prevesti pa pokrenuti iz naredbenog retka pozivima:

```
mvn compile
java -cp target/classes ispit.Pokretac
```

Jednom kad ste to napravili, krenite na rješavanje zadataka.

Zadatak 1.

Napišite programski kod upravitelja razmještajem `ExamLayoutManager`.



Površina kontejnera

Podjela pod upraviteljem razmještaja

Upravitelj razmještaja površinu dijeli u tri područja, kako je prikazano na slici. Upravitelj kroz konstruktor dobiva jedan parametar: postotak (cijeli broj između 10 i 90), a ima i setter za taj parametar koji omogućava da se isti izmijeni (trebat ćete ga kasnije). Pokušaj postavljanja na vrijednost manju od 10 ili veću od 90 treba baciti prikladnu iznimku.

Kako je prikazano na slici, područje 1 je široko koliko je širina samog kontejnera, dok je visina jednaka zadanom postotku visine kontejnera (na prikazanoj slici to je poprilici 33%). Područja 2 i 3 pune ostatak kontejnera. Pri tome je širina područja 2 ponovno jednaka zadanom postotku širine kontejnera (u primjeru je to, dakle, 33% širine), a područje 3 puni ostatak.

Važno: pri implementaciji napravite inačicu upravitelja razmještajem koji je modeliran najjednostavnijim sučeljem, te smisleno implementirajte samo metode nužne da ovaj zadatak proradi. U okviru ispita ne trebate kontrolirati pokušava li korisnik dodati više komponenata u isto područje i slično, pa na to ne gubite vrijeme! Dodatno, zanemarite *Insete*.

Smjestite izvorni kod ovog upravitelja u paket `zi.zad1`.

U Vaš notepad dodajte izbornik “Ispit”, pa u njega stavku “Zadatak 1.1.”. Odabirom te stavke prikazat ćete dijalog čiji je programski kod dan u nastavku. Dijalog se stvara i pokreće jednako kao i “normalan” prozor.

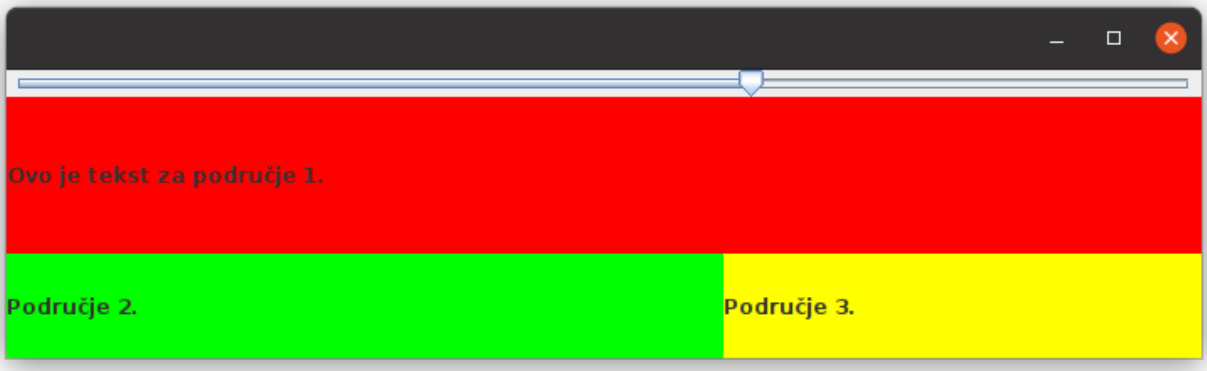
```
public class ExamZad01_1 extends JDialog {

	public ExamZad01_1() {
		setDefaultCloseOperation(DISPOSE_ON_CLOSE);
		setModal(true);

		initGUI();
		pack();
	}

	private void initGUI() {
		Container cp = getContentPane();
		ExamLayoutManager exlm = new ExamLayoutManager(20);
		cp.setLayout(exlm);

		cp.add(
		   makeLabel("Ovo je tekst za područje 1.", Color.RED),
		   ExamLayoutManager.AREA1);
		cp.add(
		   makeLabel("Područje 2.", Color.GREEN),
		   ExamLayoutManager.AREA2);
		cp.add(
		   makeLabel("Područje 3.", Color.YELLOW),
		   ExamLayoutManager.AREA3);
	}

	private Component makeLabel(String txt, Color col) {
		JLabel lab = new JLabel(txt);
		lab.setOpaque(true);
		lab.setBackground(col);
		return lab;
	}
}
```

Osigurajte da se pokretanjem ovog programa (bez izmjena programskog koda dijaloga) dobiva korektan razmještaj.

Ako se prozoru promijene dimenzije, komponente se trebaju prikladno razmještati u skladu sa zadanim postotkom.

Napomena: legalno je da je u kontejner dodan i neki podskup ovih komponenata; ako za neko područje nije definirana komponenta, to područje naprosto ostaje prazno (ali i dalje čuva svoju površinu – preostale se komponente, ako postoje, ne razmještaju po njemu). Primjerice, bez komponenata za područja 1 i 2, razmještaj bi izgledao kao na slici u nastavku.

Sada napravite kopiju razreda `ExamZad01_1` u `ExamZad01_2`, i u tom novom razredu doradite programski kod tako na vrhu prikazuje "kliznik" (JSlider) koji korisniku omogućava da odabire postotak u intervalu od 10 do 90. Kako korisnik pomiče kliznik i odabire nove postotke, razmještaj komponenata u prozoru se treba automatski ažurirati. Slika u nastavku prikazuje situaciju kada je kliznik na okvirno 60%. Važno: kao odgovor na promjenu položaja kliznika Vaš kod smije nad upraviteljem isključivo pozvati metodu (setter) za postavljanje postotka – ne smije izravno pozivati metodu upravitelja koja provodi raspoređivanje (niti to smije setter učiniti). Swing ima predviđen mehanizam kako ga obavijestiti da razmještaji više ne valjaju, pa da pokrene postupke raspoređivanja!



U Vaš notepad u izbornik "`Ispit`", dodajte stavku "`Zadatak 1.2.`". Odabirom te stavke prikazat ćete ovaj novi dijalog.

Zadatak 2.

Napravite vlastitu Swing komponentu koja u konstruktoru dobiva listu prirodnih brojeva, te na svojoj površini crta stupčasti dijagram koji ima onoliko stupaca koliko je u listi brojeva. Visina svakog stupca treba odgovarati udjelu svakog od brojeva u sumi svih brojeva. Konkretno, ako se u konstruktoru preda lista brojeva {3,1,2,0}, suma je 3+1+2+0=6. Komponenta treba prikazati 4 stupca (pa je stoga svaki širine koja odgovara širini komponente / 4). Npr., visina prvog stupca treba biti 3/6=0.5 odnosno 50% visine same komponente, visina trećeg 2/6=0.33 odnosno 33% visine komponente i slično. Stupci međusobno mogu (a i ne moraju biti odvojeni). Ako su svi brojevi u listi 0 pa se ne mogu računati udjeli, postupiti kao da je visina svih stupaca 0%.

Važno: crtate samo stupce – ne ispisujete nikakav tekst, ne crtate osi i slično. Neka su boje stupaca redom zelena, crvena, zelena, crvena, ..., a površina komponente bijela.

U Vaš notepad u izbornik "`Ispit`", dodajte stavku "`Zadatak 2.`". Odabirom te stavke treba se otvoriti novi dijalog koji će prikazati ovu komponentu na čitavoj površini. Kao brojeve trebate poslati duljine svih otvorenih dokumenata koje imate u notepadu.

Zadatak 3.

U Vaš notepad u izbornik “`Ispit`”, dodajte stavku “`Zadatak 3.`”. Odabirom te stavke treba se otvoriti novi dijalog (ali uz `setModal(false);` tako da ga možete gledati i vratiti se u editor, a da dijalog ostane prikazan). Osigurajte da se ova stavka može aktivirati samo ako u dokumentu koji korisnik trenutno gleda postoji selektirani tekst (ako ne postoji, stavka treba biti i vizualno onemogućena); jednom kad je korisnik aktiviranjem stavke otvorio dijalog, status selekcije nema nikakvog utjecaja na rad samog dijaloga.

Dijalog mora biti modeliran kao vršni razred (dakle, ne smije biti ugniježđeni razred) te kroz konstruktor smije dobiti isključivo referencu na `MultipleDocumentModel` Vašeg editora (ne smije petljati po drugim internim stvarima/varijablama editora).

Dijalog treba prikazivati listu koja ima onoliko elemenata koliko editor ima otvorenih dokumenata, a *i*-ta stavka treba prikazivati stazu *i*-tog dokumenta (ili neki specifičan tekst ako dokument još nije snimljen).

Ovaj pogled treba biti živ, u smislu da ako korisnik otvara ili zatvara nove dokumente, ili snimi dokument pa isti “dobije” stazu, i u ovoj listi se ažuriraju informacije.

U listi se automatski treba ažurirati i selekcija, tako da uvijek bude selektirana stavka koja odgovara dokumentu koji korisnik trenutno uređuje.