1. domaća zadaća; OPRPP2

Ako još niste, napravite novi radni prostor (engl. *workspace*) u okolini Eclipse. U njemu potom napravite prazan Maven projekt:

- u vršnom direktoriju radnog prostora napravite direktorij `hw01-0000000000` (zamijenite nule Vašim JMBAG-om),
- u tom novom direktoriju oformite Mavenov projekt `oprpp2.jmbag0000000000:hw01-0000000000` (zamijenite nule Vašim JMBAG-om),
- u projekt dodajte ovisnost prema biblioteci `junit` pa
- importajte projekt u Eclipse.

Sada možete nastaviti s rješavanjem zadataka.

Priprema

U okviru prva dva zadatka napravit ćete paralelzaciju generiranja fraktala koji ste na kolegiju OPRPP1 riješavali u okviru 6. domaće zadaće. Ako niste bili na tom kolegiju, pošaljite mi e-mail (što prije) pa ću Vam poslati uputu za tu domaću zadaću, kako biste mogli pripremiti rješenje dijela zadaće koji je potreban za rješavanje ovog zadatka.

Da biste mogli zadatak riješiti ispravno, napravio sam manju izmjenu u sučelju `IFractalProducer` koje modelira objekte čija je zadaća "proizvodnja" (odnosno generiranje) fraktala. Sučelje je opremljeno s dvije dodatne metode `setup` i `close` te sada izgledao ovako:

```
public interface IFractalProducer extends AutoCloseable {
	public void setup();

	public void produce(
		double reMin, double reMax, double imMin, double imMax,
		int width, int height, long requestNo,
		IFractalResultObserver observer, AtomicBoolean cancel
	);

	public void close();
}
```

Objektu koji je primjerak razreda koji implementira ovo sučelje garantira se da će se najprije jednom pozvati metoda `setup`, kako bi objekt mogao pripremiti sve potrebne resurse koje će koristiti pri generiranju fraktala. Nakon što ta metoda uspješno završi, pozivat će se metoda `produce` svaki puta kada treba izgenerirati novu sliku fraktala. Ta metoda mora biti višedretveno sigurna, jer je moguće i da se pozove iz jedne dretve, dok prethodni poziv napravljen iz druge dretve još uvijek nije završen. Konačno, jednom kada objekt za generiranje fraktala više nije potreban (primjerice, kad se zatvara prozor), nad njim će se pozvati metoda `close`, čija je zadaća osloboditi eventualno zauzete resurse. Objektu se garantira da se metoda `close` neće pozvati tako dugo dok se izvodi barem jedan poziv metode `produce`.

Na Ferka u repozitorij uz ovu zadaću postavio sam novogeneriranu biblioteku te njezin javadoc:

- `fractal-viewer-1.1.jar`
- `fractal-viewer-1.1-javadoc.jar`

Postavite u Vaš lokalni Mavenov repozitorij ovu biblioteku zajedno s javadocom pod koordinatama: `hr.fer.zemris.java.fractals:fractal-viewer:1.1`
te potom dodajte ovisnost o ovoj biblioteci u Vaš novostvoreni projekt. Važno: prilikom predaje domaće zadaće, u ZIP-arhivu **ne pakirate** ovu biblioteku niti njezin javadoc.

U Vaš projekt sada iz šeste domaće zadaće s kolegija OPRPP1 prekopirajte razrede koji su bili relevantni za postupak generiranja fraktala (primjerice, modele kompleksnih brojeva odnosno polinoma).

Problem 1.

Napišite glavni program `hr.fer.zemris.java.fractals.NewtonP1`. Program korisnika treba pitati da unese nultočke polinoma (“korijene”) prema sintaksi iz izvornog zadatka; ovdje u nastavku ponavljam taj dio upute.

Nakon što korisnik unese nultočke, program treba pokrenuti preglednik fraktala uz prikladnu implementaciju objekta zaduženog za provođenje izračuna koji je modeliran sučeljem `IFractalProducer`. Taj objekt u ovom zadatku za paralelizaciju treba koristiti okruženje `Executors`, odnosno prikladnu implementaciju koja koristi fiksni broj radnika. Preglednik fraktala će taj objekt po potrebi tražiti da provede sve proračune i na temelju vraćenih rezultata prikazati sliku fraktala u prozoru.

Primjer interakcije s korisnikom prikazan je u nastavku.

```
C:\somepath> java -cp sve-što-treba hr.fer.zemris.java.fractals.NewtonP1
Welcome to Newton-Raphson iteration-based fractal viewer.
Please enter at least two roots, one root per line. Enter 'done' when done.
Root 1> 1
Root 2> -1 + i0
Root 3> i
Root 4> 0 - i1
Root 5> done
Image of fractal will appear shortly. Thank you.
```

(korisnikov unos prikazan je crvenom bojom)

Oblik unosa kompleksnih brojeva koje treba podržati su unosi oblika *a+ib* odnosno *a-ib* gdje se dijelovi koji su 0 mogu ispustiti (ali ne oba istovremeno). Primjerice, nula se može zadati kao 0, i0, 0+i0, 0-i0. Ako u unosu postoji „i” ali nema *b*-dijela, pretpostaviti da je b=1. To znači da su i sljedeći unosi ispravni: i, 4+i, 5-i, -i.

Prilikom pokretanja programa treba dozvoliti i zadavanje parametara kako je opisano u nastavku, odnosno kako ilustrira sljedeći poziv (zbog čitljivosti sam uklonio dio koji specificira parametar *classpath*, i koji ćete pri pokretanju morati zadati):

```
java hr.fer.zemris.java.fractals.NewtonP1 --workers=2 --tracks=10
```

Parametar `--workers=N` određuje broj dretvi (radnika) koje se koriste za paralelizaciju. Treba podržati i kraći oblik specifikacije ovog parametra: `-w N`. Ako parametar nije zadan, kao vrijednost treba koristiti broj procesora koji su na raspolaganju Vašem procesu (pogledajte: `Runtime.getRuntime()` i način na koji se može doznati koliko imate procesora na raspolaganju).

Broj traka na koji ćete razložiti posao generiranja fraktala zadaje se parametrom `--tracks=K` (ili kraće: `-t K`). Minimalni prihvatljivi `K` je 1. Ako korisnik zada `K` koji je veći od broja redaka slike, kao „efektivni” broj traka treba koristiti broj redaka slike. Ako se parametar ne zada, koristite
4 * `numberOfAvailableProcessors`.

Uočite, svi parametri su opcionalni (jer je opisano koje su im pretpostavljene vrijednosti); također, svaki se može zadati kroz dvije različite sitakse. Poredak kojim ih korisnik zada nije bitan (odnosno može biti proizvoljan). Konačno, greška je zadati isti parametar više od jednom (čak i ako je vrijednost ista).

Pri pokretanju pograma na zaslon ispišite koliko procesora stvarno koristite. Svaki puta kada započnete postupak generiranja novog fraktala na zaslon ispišite koliko traka koristite.

Problem 2.

Napišite glavni program `hr.fer.zemris.java.fractals.NewtonP2`. Komunikacija programa s korisnikom je ista kao u prethodnom zadatku (vezano za način unosa nultočki). Objekt koji obavlja generiranje fraktala u ovom zadatku treba koristiti `ForkJoinPool` i modeliranje posla koje je prirodno za taj razred.

Program treba podržati samo jedan opcionalni parametar koji se može zadati kroz naredbeni redak:
`--mintracks=K` (ili kraće: `-m K`)
koji predstavlja minimalni broj redaka posla koji se više neće rekurzivno dekomponirati već će se izračunati slijedno. Ako se parametar ne zada, pretpostavljena mu je vrijednost 16.

Važno. Smijete se konzultirati s Vašim kolegama i razmijenjivati ideje od trenutka kada ste pročitali uputu pa sve do trenutka kada krenete rješavati zadatake. Od trenutka kada napišete prvu liniju koda rješenja daljnja komunikacija je zabranjena. Naravno, u slučaju bilo kakvih nejasnoća, mene uvijek možete kontaktirati.

Dokumentirajte Vaš kod. Projekt i sav izvorni kod mora biti pohranjen uporabom kodne stranice UTF-8.

Kada ste završili sa zadaćom, zapakirajte projekt u ZIP-arhivu imena `hw01-0000000000.zip` (zamijenite nule Vašim stvarnim JMBAG-om). U ZIP-arhivu **ne pakirate** datoteke koje nastaju prevođenjem (tipa: direktorij `target`). Arhivu na Ferka uploadajte **prije** isteka roka. Ako ne stignete predati rješenje do tog roka, na zasebnom mjestu u Ferku bit će omogućena zakašnjela predaja koja će se prihvaćati još najviše 72 sata nakon izvornog roka (uvjeti su specificirani u početnoj prezentaciji).

Nemojte zaboraviti zaključati Vaš upload jer ga inače nećemo prihvatiti.