Neizrazito, evolucijsko i neuroračunarstvo: zadatak 1

Tema ovog zadatka su neizraziti skupovi i operacije nad njima. U nastavku je prikazan objektni model rješenja, bez konkretne implementacije. Niste dužni napraviti *baš takav model*: ono što je važno jest da pogledate demonstracijske primjere i napravite rješenje koje nudi sličnu funkcionalnost. Struktura rješenja koja je ovdje prikazana predstavlja preporučenu organizaciju rješenja. U okviru drugog zadatka prikazano ćete rješenje nadopuniti potporom za neizrazite relacije.

Kako bi uputa bila što manje ovisna o konkretnom jeziku, uz primjere dani su i dijagrami razreda. Na temelju svih tih informacija ne biste smjeli imati problema s izradom rješenja u bilo kojem od popularnijih jezika (C++, C#, Java, Python).

Cjelokupno programsko rješenje (izvorni kod i te izvršni kod prikazanih demonstracijskih primjera) potrebno je uploadati na Ferka pod 1. domaću zadaću i zaključati do isteka roka.

Ovaj zadatak sastoji se od tri cjeline koje se nadograđuju:

1. modeliranje domene neizrazitog skupa,
2. modeliranje neizrazitog skupa te
3. modeliranje operacija nad neizrazitim skupom.

1. Modeliranje domene neizrazitog skupa

Kako ne bismo previše zakomplicirali rješavanje, pretpostavit ćemo da neizrazite skupove definiramo nad kontinuiranim podskupom cijelih brojeva odnosno nad kartezijevim produktima takvih skupova. U najjednostavnijem slučaju domena će biti primjerice podskup cijelih brojeva od -12 do +7. U složenijem slučaju, elementi domene mogu biti uređeni parovi cijelih brojeva poput dvojki (2,1), trojki (-17,21,3) i slično (nema unaprijed definirane fiksne granice na broj komponenata, osim naravno raspoložive memorije).

Evo primjera definicije i uporabe domena.

```
public class Domene {

	public static void main(String[] args) {
		IDomain d1 = Domain.intRange(0, 5);  // {0,1,2,3,4}
		Debug.print(d1, "Elementi domene d1:");

		IDomain d2 = Domain.intRange(0, 3);  // {0,1,2}
		Debug.print(d2, "Elementi domene d2:");

		IDomain d3 = Domain.combine(d1, d2);
		Debug.print(d3, "Elementi domene d3:");

		System.out.println(d3.elementForIndex(0));
		System.out.println(d3.elementForIndex(5));
		System.out.println(d3.elementForIndex(14));
		System.out.println(d3.indexOfElement(DomainElement.of(4,1)));
	}

}
```

Očekivani ispis programa je:

```
Elementi domene d1:
Element domene: 0
Element domene: 1
Element domene: 2
Element domene: 3
Element domene: 4
Kardinalitet domene je: 5

Elementi domene d2:
Element domene: 0
Element domene: 1
Element domene: 2
Kardinalitet domene je: 3

Elementi domene d3:
Element domene: (0,0)
Element domene: (0,1)
Element domene: (0,2)
Element domene: (1,0)
Element domene: (1,1)
Element domene: (1,2)
Element domene: (2,0)
Element domene: (2,1)
Element domene: (2,2)
Element domene: (3,0)
Element domene: (3,1)
Element domene: (3,2)
Element domene: (4,0)
Element domene: (4,1)
Element domene: (4,2)
Kardinalitet domene je: 15

(0,0)
(1,2)
(4,2)
13
```

Kako je domena sastavljena od elemenata nad kojima postoji uređaj, objekt koji predstavlja domenu nudi dvije pomoćne metode: jednu koja prima redni broj elementa domene i vraća ga (pogledajte ispis domene d3: nulti element domene je uređeni par (0,0), prvi element domene je uređeni par (0,1), i tako dalje sve do 14. elementa koji je uređeni par (4,2)), odnosno drugu koja prima element domene i vraća njegov redni broj.

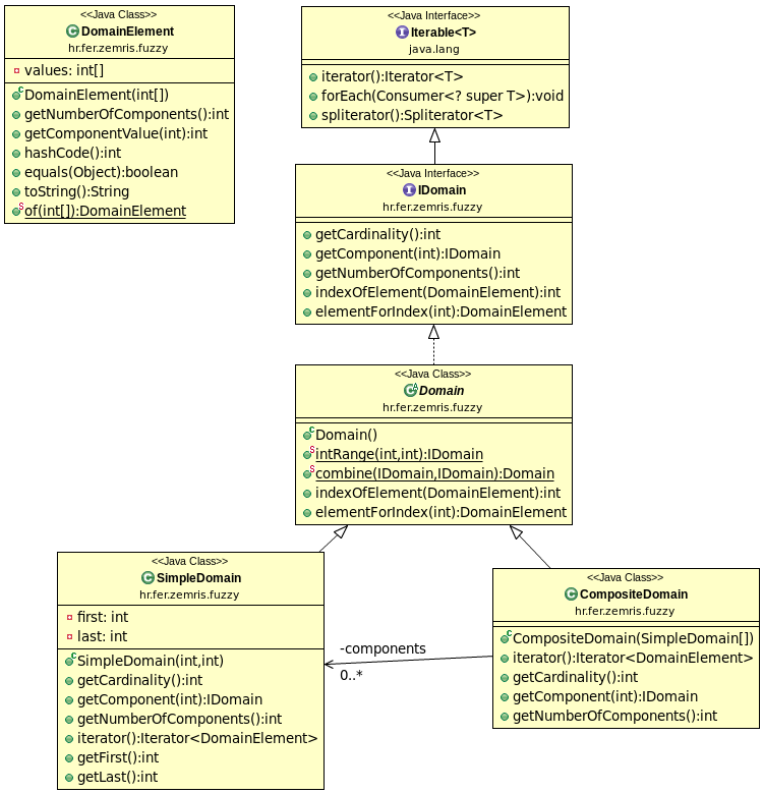
Element domene modeliran je razredom **`DomainElement`**; primjerci tog razreda predstavljaju konkretne uređene *n*-torke cijelih brojeva (i kako smo u objektnoj paradigmi, razred definira i kako se dva takva elementa uspoređuju, kako im se računa sažetak te kako se pretvaraju u string).

U pomoćni razred **`Debug`** smjestili smo metodu za ispis elemenata domene:

```
public static void print(IDomain domain, String headingText) {
	if(headingText!=null) {
		System.out.println(headingText);
	}
	for(DomainElement e : domain) {
		System.out.println("Element domene: " + e);
	}
	System.out.println("Kardinalitet domene je: " + domain.getCardinality());
	System.out.println();
}
```

Kao što je vidljivo iz ovog primjera, domena mora ponuditi mogućnost prolaska kroz sve njezine elemente kao i mogućnost dohvata kardinaliteta domene.

Prijedlog organizacije koda dan je na sljedećem dijagramu razreda.



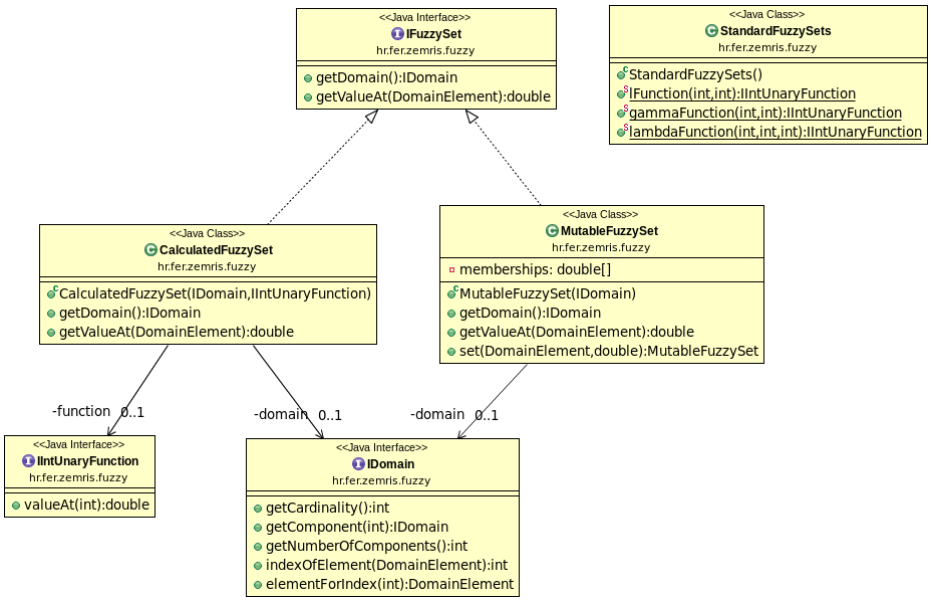
Razred **`SimpleDomain`** predstavlja jednostavnu domenu (u smislu podskupa cijelih brojeva; elementi su "degradirane" *n*-torke gdje je *n*=1). Kroz konstruktor prima granice intervala pri čemu je prva uključiva a druga isključiva. Razred **`CompositeDomain`** predstavlja kartezijev produkt jednostavnih domena (elementi uređene *n*-torke gdje je *n*>1).

Sučelje **`IDomain`** predstavlja model domene i popisuje svu funkcionalnost domene. Metoda **`getNumberOfComponents()`** vraća broj jednostavnih domena koje sudjeluju u kartezijevom skupu a **`getComponent(i)`** vraća *i*-tu komponentu tog kartezijevog skupa. Jednostavna domena vraća 1 i samu sebe.

Pogledajte prvi demonstracijski program i uočite kako korisnik nigdje ne treba direktno raditi s ovim izvedenim razredima (**`Simple`**/**`CompositeDomain`**) već svu funkcionalnost ostvaruje kroz apstraktniji razred **`Domain`** (koji je svjestan ovih implementacija i maskira ih kroz statičke metode **`intRange`**/**`combine`**).

2. Model neizrazitog skupa

Neizrazit skup definiran je funkcijom pripadnosti nad elementima domene. Ta je definicija izravno modelirana sučeljem **`IFuzzySet`**.



Imamo dvije implementacije. Razred **`MutableFuzzySet`** vrijednosti funkcije pripadnosti čuva u internom polju **`double`**-ova. Već smo rekli da svi elementi domene imaju svoj jedinstveni redni broj. U metodi **`MutableFuzzySet.getValueAt(`***e***`)`** naprosto pitamo domenu koji je redni broj od elementa *e* i na to mjesto pogledamo u polju. Metoda **`set(`***e*`,`*mu*`)`** postavlja vrijednost funkcije pripadnosti za zadani element *e*.
Implementacija **`CalculatedFuzzySet`** predstavlja implementaciju koja dobiva domenu te referencu na funkciju na temelju koje se određuje mjera kojom elementi pripadaju neizrazitom skupu (no to se nigdje ne pamti već svaki puta računa na zahtjev). Ova funkcija modelirana je nad cijelim brojevima sučeljem **`IIntUnaryFunction`**. Ideja je funkcije poput *l*-funkcije, *lambda*-funkcije i *gamma*-funkcije iz predavanja razvući nad rednim brojevima elemenata domene. Pomoćni razred **`StandardFuzzySets`** nudi tri metode namijenjene upravo stvaranju ovih funkcija. Primjer uporabe dan je u nastavku.

```
public class Primjer1 {

	public static void main(String[] args) {

		IDomain d = Domain.intRange(0, 11); // {0,1,...,10}

		IFuzzySet set1 = new MutableFuzzySet(d)
			.set(DomainElement.of(0), 1.0)
			.set(DomainElement.of(1), 0.8)
			.set(DomainElement.of(2), 0.6)
			.set(DomainElement.of(3), 0.4)
			.set(DomainElement.of(4), 0.2);
		Debug.print(set1, "Set1:");
```

```
            IDomain d2 = Domain.intRange(-5, 6); // {-5,-4,...,4,5}
            IFuzzySet set2 = new CalculatedFuzzySet(
                        d2,
                        StandardFuzzySets.lambdaFunction(
                              d2.indexOfElement(DomainElement.of(-4)),
                              d2.indexOfElement(DomainElement.of( 0)),
                              d2.indexOfElement(DomainElement.of( 4))
                        )
            );
            Debug.print(set2, "Set2:");
      }

}
```

Očekivani ispis programa je:

```
Set1:
d(0)=1.000000
d(1)=0.800000
d(2)=0.600000
d(3)=0.400000
d(4)=0.200000
d(5)=0.000000
d(6)=0.000000
d(7)=0.000000
d(8)=0.000000
d(9)=0.000000
d(10)=0.000000

Set2:
d(-5)=0.000000
d(-4)=0.000000
d(-3)=0.250000
d(-2)=0.500000
d(-1)=0.750000
d(0)=1.000000
d(1)=0.750000
d(2)=0.500000
d(3)=0.250000
d(4)=0.000000
d(5)=0.000000
```

Uočite da smo u razred **`Debug`** dodali prikladnu metodu za ispis neizrazitog skupa.

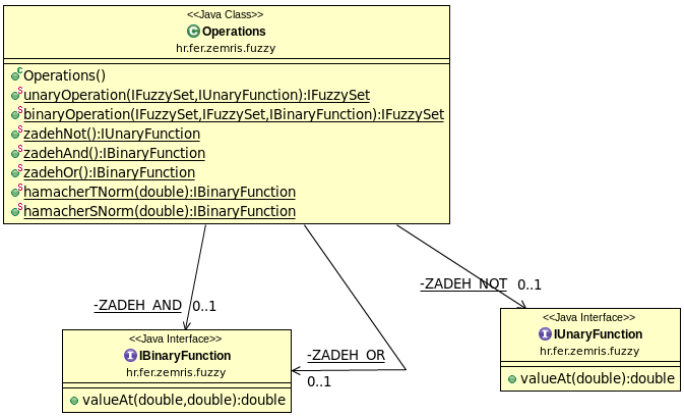
3. Modeliranje operacija nad neizrazitim skupovima

Nad neizrazitim skupovima tipično obavljamo unarne operacije (tražimo novi neizraziti skup koji je primjerice komplement zadanog neizrazitog skupa) te binarne operacije (tražimo novi neizraziti skup koji je unija ili presjek dvaju drugih neizrazitih skupova *definiranih nad istom domenom*).

U tu svrhu definirat ćemo dva pomoćna sučelja: model unarne i binarne funkcije: **`IUnaryFunction`** te **`IBinaryFunction`**.

Potom ćemo definirati nekoliko takvih funkcija (primjerice, klasična funkcija *negacije* odgovara preslikavanju $x \rightarrow 1\text{-}x$, Zadehova *t*-norma odgovara preslikavanju $(a,b) \rightarrow \min(a,b)$, itd). Ove funkcije i metode koje ih stvaraju smjestili smo u razred **`Operations`**.

Dijagram razreda prikazan je u nastavku.



Primjer uporabe:

```
public class Primjer2 {

	public static void main(String[] args) {

		IDomain d = Domain.intRange(0, 11);

		IFuzzySet set1 = new MutableFuzzySet(d)
			.set(DomainElement.of(0), 1.0)
			.set(DomainElement.of(1), 0.8)
			.set(DomainElement.of(2), 0.6)
			.set(DomainElement.of(3), 0.4)
			.set(DomainElement.of(4), 0.2);
		Debug.print(set1, "Set1:");

		IFuzzySet notSet1 = Operations.unaryOperation(
					set1, Operations.zadehNot());
		Debug.print(notSet1, "notSet1:");

		IFuzzySet union = Operations.binaryOperation(
					set1, notSet1, Operations.zadehOr());
		Debug.print(union, "Set1 union notSet1:");

		IFuzzySet hinters = Operations.binaryOperation(
					set1, notSet1, Operations.hamacherTNorm(1.0));
		Debug.print(hinters, "Set1 intersection with notSet1 using
parameterised Hamacher T norm with parameter 1.0:");
	}

}
```

bi morao rezultirati sljedećim ispisom:

```
Set1:
d(0)=1.000000
d(1)=0.800000
d(2)=0.600000
d(3)=0.400000
d(4)=0.200000
d(5)=0.000000
d(6)=0.000000
d(7)=0.000000
d(8)=0.000000
d(9)=0.000000
d(10)=0.000000
```

```
notSet1:
d(0)=0.000000
d(1)=0.200000
d(2)=0.400000
d(3)=0.600000
d(4)=0.800000
d(5)=1.000000
d(6)=1.000000
d(7)=1.000000
d(8)=1.000000
d(9)=1.000000
d(10)=1.000000

Set1 union notSet1:
d(0)=1.000000
d(1)=0.800000
d(2)=0.600000
d(3)=0.600000
d(4)=0.800000
d(5)=1.000000
d(6)=1.000000
d(7)=1.000000
d(8)=1.000000
d(9)=1.000000
d(10)=1.000000

Set1 intersection with notSet1 using parameterised Hamacher T norm with parameter 1.0:
d(0)=0.000000
d(1)=0.160000
d(2)=0.240000
d(3)=0.240000
d(4)=0.160000
d(5)=0.000000
d(6)=0.000000
d(7)=0.000000
d(8)=0.000000
d(9)=0.000000
d(10)=0.000000
```

Prilikom izrade Vašeg rješenja pogledajte najprije primjere uporabe i uočite koji su važni detalji koje bi i Vaše rješenje (možda na neki drugi način) trebalo sadržavati. U ovom posljednjem primjeru to bi, primjerice, bila razdvojenost neizrazitog skupa i operacije kojom je definiran. Želimo li u program dodati Yagerovu s- i t-normu, sve što je potrebno je napisati njihove implementacije – ništa ne trebamo mijenjati u dijelu koda koji se bavi modeliranjem neizrazitih skupova.