11. Набројиви (енумерисани) тип

Набројиви тип (енг. enumerated type/enum) је специјални референтни тип података који узима вредности из унапред одређеног скупа могућих вредности. Све вредности које може узети променљива набројивог типа су познате већ у фази компилације програма. На пример, постоји 7 дана у недељи, 12 месеци у години и јасно је да нема смисла креирање још неког дана или месеца у току извршавања програма. Потреба за оваквим типом података је постојала и пре него што је подршка за набројиве типове додата у језик и она се обично решавала креирањем целобројних статичких поља. На пример, за дане у недељи и месеце у години би то изгледало попут:

```
public static final int PONEDELJAK = 0;
public static final int UTORAK = 1;
...
public static final int JANUAR = 0;
public static final int FEBRUAR = 1;
...
```

Овај приступ, међутим, не омогућава заштиту од мешања различитих типова података. На пример, следеће поређење би било потпуно легитимно преведено иако је семантика употребе врло упитна.

```
int dan = JANUAR;
if(dan==PONEDELJAK){
//...
}
```

Дакле, променљивој која семантички представља дан у недељи и надаље се тако користи се додељује нешто што представља месец у години.

Јава је увела из тог разлога подршку за набројиве типове који спречавају овакве ситуације.

Пример 1. Овај пример демонстрира дефинисање набројивог типа и употребу набројиве променљиве за дане у недељи. □

```
package rs.math.oop.g11.p01.enumiDaniUNedelji;

public enum DanUNedelji{
	PONEDELJAK, UTORAK, SREDA, CETVRTAK, PETAK, SUBOTA, NEDELJA;

	public static void main(String[] args) {
		// променљиве добијају већ унапред дефинисане вредности
		DanUNedelji d1 = DanUNedelji.CETVRTAK;
		DanUNedelji d2 = DanUNedelji.UTORAK;
		DanUNedelji d3 = DanUNedelji.CETVRTAK;
		//DanUNedelji d4 = new DanUNedelji(); // није могуће
		System.out.println(d1.name());
		// аутоматска додела целобројних вредности
		System.out.println(d1.ordinal());
		// различите вредности
		System.out.println(d1==d2);
		// исте вредности - показују чак и на исту меморију
		System.out.println(d1==d3);
	}
```

```
}
```

Приликом дефинисања набројивог типа, програмер наводи списак свих вредности и тај списак се не може мењати током рада програма - ове вредности су попут унапред креираних објеката без могућности даљег позивања конструктора. Датотека у којој је дефинисан набројиви тип такође може садржати `main` методу тако да `DanUNedelji` надаље демонстрира рад са набројивим променљивама. Као што се може видети из исписа испод, набројива променљива реферише на објекат који има неколико већ уграђених атрибута попут имена и редног броја. Редни бројеви почињу од 0 и повећавају се за 1 код сваке наредне набројиве константе - тако `CETVRTAK` има додељен редни број 3. Такође треба приметити да меморија на коју реферишу набројиве променљиве заиста подсећа на објекте (што они заиста и јесу) будући да поређење променљивих `d1` и `d3` помоћу оператора `==` производи тачну вредност. Дакле, нема потребе за употребом методе `equals` како би се семантички упоредиле две набројиве променљиве.

```
CETVRTAK
3
false
true
```
■

11.1. Набројиви типови и наредба switch

Набројиви типови се често користе за прегледно представљање неких могућности унутар програмског кода па су природно повезани са контролним структурама. Поред употребе унутар `if-elseif-else` конструкција њихова употреба је подржана и контролном структуром `switch`.

Пример 2. Креирати набројиви тип за представљање месеци у току године и написати код који за дати месец и дату годину враћа број дана у том месецу. Унос жељеног месеца и године се уноси путем аргумената командне линије. □

```
package rs.math.oop.g11.p02.enumiBrojDanaUMesecu;

public enum MesecKalendarski {
	JANUAR, FEBRUAR, MART, APRIL, MAJ, JUN,
	JUL, AVGUST, SEPTEMBAR, OKTOBAR, NOVEMBAR, DECEMBAR;

	public static void main(String[] args) {
		if (args.length != 2) {
			System.err.println("Употреба: java <назив месеца> <година>");
			System.exit(-1);
		}
		// производи изузетак ако унесемо нешто што се не налази у списку назива
		// набројивих типова
		try {
			MesecKalendarski mesec =
					MesecKalendarski.valueOf(args[0].toUpperCase());
			int godina = Integer.valueOf(args[1]);
			int brojDana = 0;
			switch (mesec) {
			case APRIL:case JUN:case SEPTEMBAR:case NOVEMBAR:
				brojDana = 30;
				break;
```

```
                        case JANUAR:case MART: case MAJ: case JUL:
                                case AVGUST:case OKTOBAR: case DECEMBAR:
                                        brojDana=31;
                                        break;
                                case FEBRUAR:
                                        if ((godina % 4 == 0 && godina % 100 != 0)
                                                        || (godina % 400 == 0))
                                                brojDana = 29;
                                        else
                                                brojDana = 28;
                                        break;
                                }
                                System.out.printf("Број дана у месецу %s године %d је %d.",
                                                mesec, godina, brojDana);
                        } catch (IllegalArgumentException ex) {
                                System.err.printf("Грешка при парсирању месеца са називом %s.",
                                                args[1]);
                                System.err.println(ex.getMessage());
                        }
                }
        }
```

Након што се учита кориснички унос са називом месеца, на основу тог уноса се дохвата одговарајућа набројива вредност (константа) помоћу статичке методе `valueOf`. У случају да је дат текст који када се пребаци у велика слова не одговара називу ниједне доступне набројиве вредности избацује се изузетак типа `IllegalArgumentException`. Ако је све у реду, програм наставља даље и помоћу наредбе switch која омогућава прегледан приказ могућности, одређује се број дана за сваки месец. За месец фебруар постоји додатно разматрање у вези преступне године, јер свака четврта је преступна осим оних које су дељиве са 100 и притом нису дељиве са 400. Приказ из излаза програма за унете аргументе “februar 2000” (латинични текст због назива набројивих вредности) је:

```
Број дана у месецу FEBRUAR године 2000 је 29.
```

Ако би унос био неодговарајући, тј. унето нешто што не постоји у списку назива, десила би се грешка у виду горепоменутог изузетка, нпр. за “test 2002”:

```
Грешка при парсирању месеца са називом 2002.No enum constant
rs.math.oop.g11.p02.enumiBrojDanaUMesecu.MesecKalendarski.TEST
```

11.2. Обогаћивање набројивих типова, конструктори и методе

Набројиви типови у Јави могу носити и додатне информације поред оних имплицитно креираних попут имена и редног броја. Ово се постиже додавањем нових поља, креирањем нових метода или превазилажењем постојећих као и додавањем нових експлицитних конструктора.

Пример 3. Креирати набројиви тип за представљање планета Сунчевог система уз додатне информације о маси и пречнику сваке планете и исписати све ове информације за сваку планету. Додатно, омогућити да се израчуна тежина тела на површини свих планета при чему се као аргумент командне линије задаје маса тела. □

```
package rs.math.oop.g11.p03.enumiPlaneteSuncevogSistema;
```

```
public enum PlanetaSuncevogSistema {
    MERKUR (3.303e+23, 2.4397e6),
    VENERA   (4.869e+24, 6.0518e6),
    ZEMLJA   (5.976e+24, 6.37814e6),
    MARS    (6.421e+23, 3.3972e6),
    JUPITER (1.9e+27,   7.1492e7),
    SATURN  (5.688e+26, 6.0268e7),
    URAN  (8.686e+25, 2.5559e7),
    NEPTUN (1.024e+26, 2.4746e7);

    private final double masa;   // kg
    private final double precnik; // m

    PlanetaSuncevogSistema(double masa, double precnik) {
        this.masa = masa;
        this.precnik = precnik;
    }

    double masa() { return masa; }
    double precnik() { return precnik; }

    // универзална гравитациона константа (m3 kg-1 s-2)
    public static final double G = 6.67300E-11;

    double gravitacijaNaPovrsini() {
        return G * masa / (precnik * precnik);
    }

    double tezinaTelaNaPovrsini(double masaTela) {
        return masaTela * gravitacijaNaPovrsini();
    }

    @Override
    public String toString() {
       return String.format("%d %s %g %g",ordinal(), name(), masa, precnik);
    }

    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Употреба: java <маса тела (kg)>");
            System.exit(-1);
        }
        double masaTela = Double.parseDouble(args[0]);
        // метода values() враћа списак свих вредности за дати набројиви тип
        System.out.println("Карактеристике планета: редни број, име, маса и пречник:");
        for (PlanetaSuncevogSistema p : PlanetaSuncevogSistema.values())
              System.out.println(p);
        System.out.printf(System.lineSeparator()
                     +"Тежина тела масе %.2f на различитим планетама су%n",masaTela);
        for (PlanetaSuncevogSistema p : PlanetaSuncevogSistema.values())
           System.out.printf("%s %.2f N%n",
                             p.name(), p.tezinaTelaNaPovrsini(masaTela));
    }
}
```

Конструктор који је дефинисан унутар набројивог типа имплицитно је видљив само унутар датотеке у којој је набројиви тип и дефинисан и није га могуће позвати

експлицитно на неком другом месту осим тамо где се наводи списак набројивих вредности. Приметимо да су набројивом типу PlanetaSuncevogSistema додељене и нове методе попут `gravitacijaNaPovrsini` и `tezinaTelaNaPovrsini`, али и превазиђена је постојећа метода `toString`. За унос од 88 килограма, програм формира следећи испис.

```
Карактеристике планета: редни број, име, маса и пречник:
0 MERKUR 3.30300e+23 2.43970e+06
1 VENERA 4.86900e+24 6.05180e+06
2 ZEMLJA 5.97600e+24 6.37814e+06
3 MARS 6.42100e+23 3.39720e+06
4 JUPITER 1.90000e+27 7.14920e+07
5 SATURN 5.68800e+26 6.02680e+07
6 URAN 8.68600e+25 2.55590e+07
7 NEPTUN 1.02400e+26 2.47460e+07

Тежина тела масе 88.00 на различитим планетама су
MERKUR 325.87 N
VENERA 780.68 N
ZEMLJA 862.63 N
MARS 326.71 N
JUPITER 2182.94 N
SATURN 919.58 N
URAN 780.79 N
NEPTUN 981.96 N■
```

У неким ситуацијама, корисно је набројиви тип обогатити и методом која ће бити реализована различито за различите вредности - а не исто као за досад уведене додатне методе попут метода `masa`, `precnik`, `tezinaTelaNaPovrsini` и `gravitacijaNaPovrsini`. Приметите да је овај механизам могао да послужи у примеру са бројем дана у месецу где је уместо једне централизоване методе свака конкретна набројива вредност могла да има засебну реализацију методе која за дати параметар календарска година враћа број дана у датом месецу. Ипак, ово понашање ћемо илустровати кроз нови пример.

Пример 4. Креирати набројиви тип за представљање основних аритметичких операција и њихову реализацију. □

```
package rs.math.oop.g11.p04.enumiAritmetickeOperacije;

public enum AritmetickaOperacija {
	PLUS("+") {
		public double izracunaj(double x, double y) {
			return x + y;
		}
	},
	MINUS("-") {
		public double izracunaj(double x, double y) {
			return x - y;
		}
	},
	PUTA("*") {
		public double izracunaj(double x, double y) {
			return x * y;
		}
	},
	PODELJENO("/") {
```

```
                public double izracunaj(double x, double y) {
                        return x / y;
                }
        };

        private final String oznaka;

        AritmetickaOperacija(String oznaka) {
                this.oznaka = oznaka;
        }

        @Override
        public String toString() {
                return oznaka;
        }

        /* абстрактна метода коју свака конкретна набројива вредност
         мора да реализује одмах при декларацији (слично анонимним класама) */
        public abstract double izracunaj(double x, double y);

        public static void main(String[] args) {
                double x = Double.parseDouble(args[0]);
                double y = Double.parseDouble(args[1]);
                for (AritmetickaOperacija op : AritmetickaOperacija.values())
                        System.out.printf("%f %s %f = %f%n", x, op, y, op.izracunaj(x, y));
        }
}
```

Видимо да је могуће декларисати абстрактну методу унутар набројивог типа, али да је и обавезно одмах реализовати те методе приликом декларације, тј. навођења вредности набројивих типова. Ово доста подсећа на рад са анонимним класама где се нека абстрактна класа или интерфејс може инстанцирати уз моментално навођење реализацију свих абстрактних метода. Испис извршавања овог примера за аргументне командне линије 23 и 10 је следећи:

```
23.000000 + 10.000000 = 33.000000
23.000000 - 10.000000 = 13.000000
23.000000 * 10.000000 = 230.000000
23.000000 / 10.000000 = 2.300000■
```

11.3. Реализација набројивог типа помоћу класе

Набројиви типови су ефективно реализовани као класе, а њихове вредности су заправо јавна статичка поља. С обзиром на своје специфично понашање и честу употребу позадина иза креирања набројивих типова је сакривена од програмера, али се она свакако може донекле симулирати кроз досада уведене концепте - не може у потпуности будући да концепт и употреба рефлексије није део овог уџбеника.

Пример 5. Реализовати претходно уведени набројиви тип `DanUNedelji` без употребе подршке за набројиве типове већ кроз употребу стандардне класе. □

```
package rs.math.oop.g11.p05.enumiDaniUNedeljiKlasa;

public final class DanUNedeljiKlasa {
        // статичка поља типа ове класе које симулирају вредности набројивог типа
```

```
	public static final DanUNedeljiKlasa PONEDELJAK = new
DanUNedeljiKlasa("PONEDELJAK");
	public static final DanUNedeljiKlasa UTORAK = new DanUNedeljiKlasa("UTORAK");
	public static final DanUNedeljiKlasa SREDA = new DanUNedeljiKlasa("SREDA");
	public static final DanUNedeljiKlasa CETVRTAK = new DanUNedeljiKlasa("CETVRTAK");
	public static final DanUNedeljiKlasa PETAK = new DanUNedeljiKlasa("PETAK");
	public static final DanUNedeljiKlasa SUBOTA = new DanUNedeljiKlasa("SUBOTA");
	public static final DanUNedeljiKlasa NEDELJA = new DanUNedeljiKlasa("NEDELJA");

	private final String naziv;
	private final int redniBroj;
	private static int brojInstanci = 0;

	private DanUNedeljiKlasa(String naziv) {
		this.naziv=naziv;
		this.redniBroj = brojInstanci;
		brojInstanci++;
	}

	public int ordinal() {
		return redniBroj;
	}

	public String name() {
		return naziv;
	}

	public static DanUNedeljiKlasa[] values() {
		return new DanUNedeljiKlasa[] { PONEDELJAK, UTORAK, SREDA,
				CETVRTAK, PETAK, SUBOTA, NEDELJA };
	}

	@Override
	public String toString() {
		return naziv;
	}

	public static void main(String[] args) {
		// сличан приступ вредностима као код набројивог типа
		DanUNedeljiKlasa d1 = DanUNedeljiKlasa.CETVRTAK;
		DanUNedeljiKlasa d2 = DanUNedeljiKlasa.UTORAK;
		DanUNedeljiKlasa d3 = DanUNedeljiKlasa.CETVRTAK;
		System.out.println(d1.name());
		// аутоматска додела целобројних вредности
		System.out.println(d1.ordinal());
		// различите вредности
		System.out.println(d1==d2);
		// исте вредности - показују чак и на исту меморију
		System.out.println(d1==d3);
		// списак вредности
		for(DanUNedeljiKlasa d: DanUNedeljiKlasa.values())
			System.out.println(d);
	}
}
```

Треба приметити да је понашање овако симулираног набројивог типа доста слично оном представљеном у Примеру 1, што демонстрира и готови идентичан садржај и даљи

испис `main` методе. Укратко, додељивање вредности набројивим променљивама изгледа исто, поређење референци одговара семантичком поређењу вредности, редни број се аутоматски додељује. Такође, реализоване су методе за испис свих вредности и превазиђена је метода `toString` - не на потпуно тачан начин, јер је за веродостојну реализацију ових метода, као што је наведено, потребна употреба концепта који се зове рефлексија.

```
CETVRTAK
3
false
true
PONEDELJAK
UTORAK
SREDA
CETVRTAK
PETAK
SUBOTA
NEDELJA■
```

11.4. Резиме

У овом поглављу уведени су неки од значајних концепата везаних за набројиве типове - постоји још неколико који су за нека напреднија разматрања попут набројивих скупова, каталога, комбиновања вишеструких набројивих типова, итд. Осећај за правилну употребу набројивих типова се стиче кроз програмерско искуство, али су неке начелне смернице да они осликавају скуп фиксираних могућности којима често могу придружене и одређене функционалности или додатне информације. Њихова фиксираност омогућава да је програмска логика потпуно свесна свих могућности те да може да одреагује на сваку од њих.

11.5. Питања и задаци