Енумерације (Хортон, прикупљено са разних места у књизи)

(пример је измењен, учинило ми се да су боје карата згоднije од комбинације дана у недељи, програма за прање машине за веш и годишњих доба)

```
enum Boja {TREF, KARO, HERC, PIK}
```

Горњим изразом се дефинише нови тип, `Boja`.
`TREF`, `KARO`, `HERC` и `PIK` се називају константама енумерације и једине су допуштене вредности за променљиве типа `Boja`.
Приметимо одсуство тачка-запете на крају дефиниције енумерације. Тачка-запета није потребна јер се ради о дефинисању новог типа.

Дефиниција енумерације може се налазити у оквиру неке друге класе, ван било ког метода, а такође може бити смештена у сопствени .java фајл чије се име поклапа са именом енумерације
(у Eclipse: нпр. File > New > Enum > да се име енумерацији > Finish)

У примеру tryEnumeration, горња дефиниција налази се унутар дефиниције тест-класе.

switch и енумерације

Променљива типа енумерације може се користити као израз који контролише `switch`.
У том случају, константе енумерације се у `case` клаузама наводе без квалификовања именом енумерације.
Уколико су као случајеви наведене све константе енумерације и `switch` није у оквиру неког метода који има повратни тип другачији од `void`, нема потребе наводити случај `default`, јер променљивој типа енумерације није могуће придружити вредност која није константа те енумерације.

Још о енумерацијама

Тип енумерације је специјална врста класе.
Када дефинишемо тип енумерације, добили смо класу која је изведена из класе `java.lang.Enum` као базне.
Константе енумерације се креирају као објекти типа те нове класе.

Објекат који одговара појединачној константи енумерације чува име константе унутар једног од својих атрибута, а наша енумерација (која је класа) наслеђује метод `toString()` од своје базне класе `Enum`.
Метод `toString()` у класи `Enum` враћа име константе енумерације. Због тога се у случају прослеђивања константе енумерације методу `System.out.println()` исписује њено име.

Објекат који одговара појединачној константи енумерације такође има целобројни атрибут.
Подразумевано се свакој константи енумерације придружује целобројна вредност различита од одговарајућих вредности придружених свим осталим константама енумерације. Вредности се придружују константама оним редом којим су оне наведене у дефиницији енумерације. Првој константи се придружује вредност 0, другој 1, итд. Одговарајућу вредност придружену константи могуће је добити помоћу метода `ordinal()`, али познавање тог броја нам, у општем случају, није неопходно.

Вредности типа енумерације могу се поредити на једнакост коришћењем метода `equals()` наслеђеног од базне класе `Enum`. Класа енумерације наслеђује од своје базне класе и метод `compareTo()` који пореди објекте типа енумерације на основу вредности њихових ординала. Овај метод враћа негативни цео број ако је вредност ординала текућег објекта мања од вредности ординала објекта прослеђеног као аргумент (текућа константа је наведена испред константе аргумента у дефиницији енумерације), 0 ако су једнаки,

а позитивну целобројну вредност ако је вредност ординала текућег објекта већа од вредности ординала објекта прослеђеног као аргумент (текућа константа је наведена иза константе аргумента у дефиницији енумерације).

Статички метод `values()` класе енумерације враћа низ који садржи све константе енумерације.
Кроз тај низ могуће је проћи коришћењем collection-based for-петље.

Додавање чланова класи енумерације

Како је енумерација класа, могуће је додавати сопствене атрибуте и методе приликом дефинисања сопственог типа енумерације. Могуће је додати и сопствене конструкторе за иницијализацију атрибута које смо увели.

Пример: enumTest

Приметите да се листа константи енумерације завршава тачка-запетом.

Иза сваке константе у листи, унутар пара облих заградица наводи се аргумент (или, у општем случају, аргументи) који се прослеђује конструктору приликом креирања објекта који одговара тој константи.

У примеру tryEnumeration није било облих заградица иза константи енумерације. Оне су се подразумевале, а подразумевао се и позив подразумеваног конструктора приликом креирања тих константи. Могао се, дакле, иза сваке константе енумерације, навести пар облих заградица, што би било еквивалентно, али не би допринело побољшању читљивости кода.
У овом примеру, када постоји један наш конструктор, компајлер не генерише подразумевани, па није могуће навести само имена константи енумерације, већ су неопходне и заградице, као и аргумент конструктора, одговарајућег типа, унутар тих заградица (конструктор који смо дефинисали очекује један аргумент типа `String`).

Уколико желимо да за неке константе енумерације изоставимо заградице, можемо дефинисати подразумевани конструктор, који атрибуту (у општем случају: атрибутима) придружује неку подразумевану вредност.

Иако смо додали сопствени конструктор, поља наслеђена од базне класе, `Enum`, која чувају име константе и вредност њеног ординала и даље ће бити исправно иницијализована. Поредак константи имплементиран методом `compareTo()` и даље ће бити одређен редоследом којим су константе наведене у дефиницији енумерације.

Није могуће да конструктор у класи енумерације буде дефинисан као `public`. Једини допуштен „модификајер“ је `private`, што ће резултовати тиме да конструктор може да се зове само унутар класе.

Могуће је предефинисати метод `toString()` у класи енумерације. Уколико се жели користити `switch` за гранање по константама, као израз који га контролише користи се референца `this`. У том случају као лабеле `case` клауза наводе се имена константи енумерације (без квалификовања именом енумерације).

Закључак: тип енумерације се може третирати као и било који други класни тип.