Paradygmaty programowania - ćwiczenia
Lista 10

Wszystkie programy mają być napisane w języku Scala.

1. a) Przepisz program z wykładu 9, str. 5 (wyjątki) na język Scala i uruchom go. Wyjaśnij komunikaty (tylko własne metody ze szczytu stosu).
   b) Przed każdą metodą, która w języku Java ma klauzulę throws, dodaj w poprzedzającym wierszu adnotację @throws(classOf[Exception]) . Wykorzystaj dekompilator Javy i zobacz, jaki efekt spowodowały te adnotacje.

2. Klasa GenericCellImm kompiluje się jako klasa inwariantna i kowariantna.

```
class GenericCellImm[T] (val x: T) {
}
//defined class GenericCellImm

class GenericCellImm[+T] (val x: T) {
}
//defined class GenericCellImm
```

Natomiast klasa GenericCellMut kompiluje się tylko jako klasa inwariantna.

```
class GenericCellMut[T] (var x: T) {
}
//defined class GenericCellMut
```

Wersja kowariantna powoduje błąd kompilacji.

```
class GenericCellMut[+T] (var x: T) {
}
//<console>:12: error: covariant type T occurs in contravariant position in type T
//of value x_=
```

Wyjaśnij a) dlaczego tak jest i b) czy można się pozbyć tego błędu.

3. Poniższa definicja powoduje błąd kompilacji.

```
abstract class Sequence[+A] {
  def append(x: Sequence[A]): Sequence[A]
}
// <console>:14: error: covariant type A occurs in contravariant position in type
// Sequence[A] of value x
```

Jak pozbyć się tego błędu (z wyjaśnieniem!).

4. Zdefiniuj generyczną inwariantną metodę copy dla kolekcji modyfikowalnych na wzór programu napisanego w Javie (wykład, str. 24).
   Wskazówka. Wykorzystaj metodę foreach z cechy scala.collection.Traversable, oraz metodę update z cechy scala.collection.mutable.Seq (patrz Scala API).

5. Zdefiniuj klasę generyczną dla kolejki niemodyfikowalnej, reprezentowanej przez parę list (patrz lista 7, zadanie 1b).
   Wskazówka. Wzoruj się na klasie dla stosu z wykładu (str. 8).