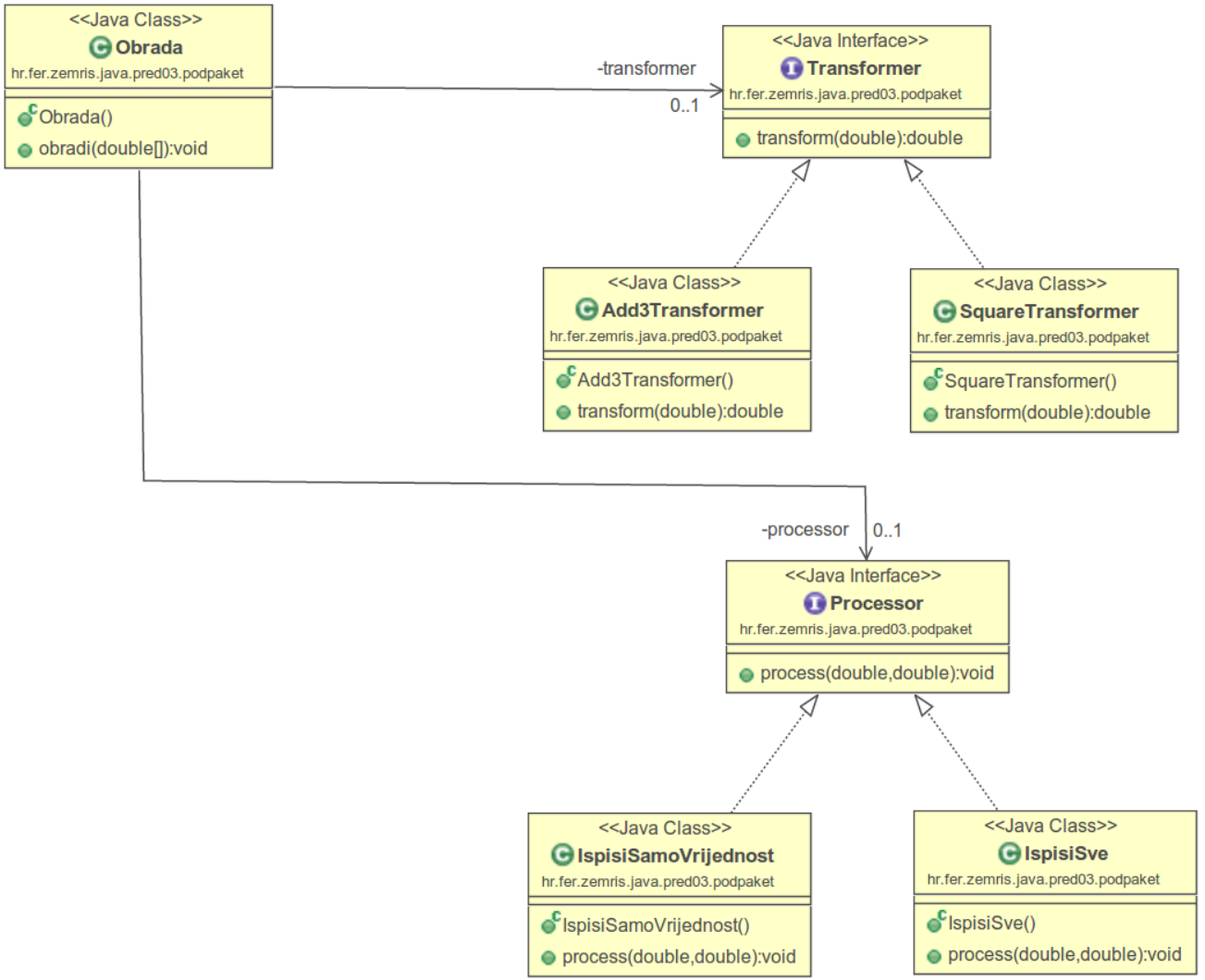
Funkcijska sučelja

Prisjetite se (3. predavanje) primjera gdje smo transformirali polje decimalnih brojeva u druge decimalne brojeve (i to samo neke).

Završili smo na oblikovnom obrascu Strategija i modeliranju objekata koji rade transformaciju te objekata koji definiraju kako se rezultat obrađuje.

Oblikovni obrazac Strategija

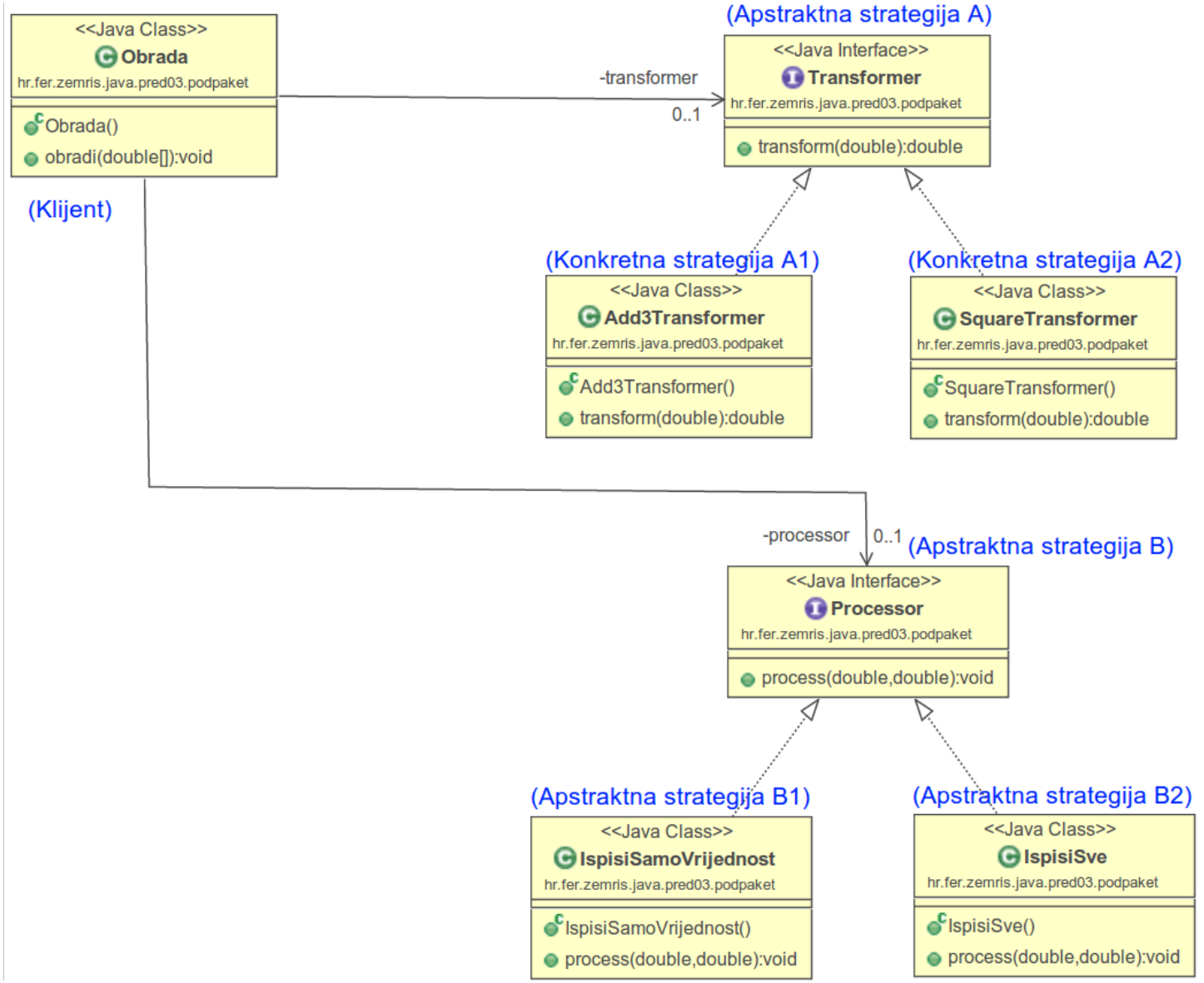


Oblikovni obrazac Strategija

- Klijent može biti razred (`Obrada`)
  - Ima članske varijable `transformer` i `processor` apstraktnog tipa kojima se konfigurira
- Klijent može biti metoda koja u tom slučaju reference preko apstraktnog tipa (to smo napravili u 3. predavanju)

```
public static void obradiBrojeve(double[] unos,
                                 Transformer t,
                                 Processor p) {
 ...
}
```

Oblikovni obrazac Strategija



Oblikovni obrazac Strategija

- Ovo rješenje omogućava pisanje generičkih metoda i usklađeno je s:
  - Načelom nadogradnje bez promjene (engl. *Open-Closed Principle*)
  - Načelom inverzije ovisnosti (engl. *Dependency Inversion*)
  - Načelom jedinstvene odgovornosti (engl. *Single Responsibility Principle*)

Operacije nad tipom `double`

- Od Jave 8 u standardne biblioteke dodan je niz funkcijskih sučelja koja služe upravo za takve stvari

DoubleConsumer

- Funkcijsko sučelje koje modelira objekte koji obrađuju jedan decimalni broj.

```
void accept(double value);
```

DoublePredicate

- Funkcijsko sučelje koje modelira objekte koji provjeravaju je li decimalni broj prihvatljiv (u logici, takve objekte nazivamo ispitnim predikatima, ili kraće, predikatima).

```
boolean test(double value);
```

DoubleUnaryOperator

- Funkcijsko sučelje koje modelira objekte koji predstavljaju unarnu operaciju nad decimalni brojem: primaju decimalni broj i pretvaraju ga u neki drugi (primjerice, negiranje, korjenovanje, potenciranje, ...).

```
double applyAsDouble(
  double operand
);
```

DoubleBinaryOperator

- Funkcijsko sučelje koje modelira objekte koji predstavljaju binarnu operaciju nad decimalnim brojevima: primaju dva decimalna broja i pretvaraju ih u neki treći (primjerice, zbrajanje, oduzimanje, množenje, ...).

```
double applyAsDouble(
  double left, double right
);
```

Općenite operacije

- Uz podršku za podskup primitivnih tipova podataka, imamo na raspolaganju i parametrizirana sučelja koja operiraju nad objektima i koja predstavljaju općenite operacije

Consumer<T>

- Funkcijsko sučelje koje predstavlja objekte koji primaju element tipa `T` i obrađuju ga.

```
void accept(T t);
```

BiConsumer<T,U>

- Funkcijsko sučelje koje predstavlja objekte koji primaju element tipa `T` i element tipa `U` i obrađuju ih.

```
void accept(T t, U u);
```

Function<T,R>

- Funkcijsko sučelje koje modelira objekte koji obavljaju funkcijsko preslikavanje: prima element tipa `T` i preslikava ga u element tipa `R`.

```
R apply(T t);
```

BiFunction<T,U,R>

- Funkcijsko sučelje koje koje modelira objekte koji obavljaju funkcijsko preslikavanje: prima jedan element tipa `T` i jedan element tipa `U` i preslikava ih u element tipa `R`.

```
R apply(T t, U u);
```

`UnaryOperator<T>`

- Funkcijsko sučelje koje koje modelira objekte koji predstavljaju općenite unarne operacije: primaju jedan element tipa `T` i preslikavaju ga u element tipa `T` (u matematičkom smislu, funkcija s `T` na `T`).

```
T apply(T t);
```

`BinaryOperator<T>`

- Funkcijsko sučelje koje koje modelira objekte koji predstavljaju općenite unarne operacije: primaju dva elementa tipa `T` i preslikavaju taj par u element tipa `T` (u matematičkom smislu, funkcija s `TxT` na `T`).

```
T apply(T t, T u);
```

Predicate<T>

- Funkcijsko sučelje koje koje modelira objekte koji predstavljaju općenite ispitne predikate: primaju jedan element tipa `T` i vraćaju je li prihvatljiv ili nije.

```
boolean test(T t);
```

`BiPredicate<T,U>`

- Funkcijsko sučelje koje koje modelira objekte koji predstavljaju općenite ispitne predikate: primaju element tipa `T` i element tipa `U` i vraćaju je li taj par prihvatljiv ili nije.

```
boolean test(T t, U u);
```

Ostalo...

- Uz navedena sučelja, postoji ih još.
- Detaljan popis može se pogledati u dokumentaciji:

  https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html

Primjer s prethodnog sata:

- Kako bismo uporabom ovih sučelja promijenili rješenje s 3. predavanja?

Dopuna:

- Rekli smo da su lambda-izrazi *konceptualno* pokrata za stvaranje primjeraka anonimnih razreda
- Rezultat su doista objekti; evo primjera:

```
DoublePredicate obj = t -> t>20.0;
```

`obj` je referenca na objekt i istu je moguće slati gdje god se očekuje `DoublePredicate`!