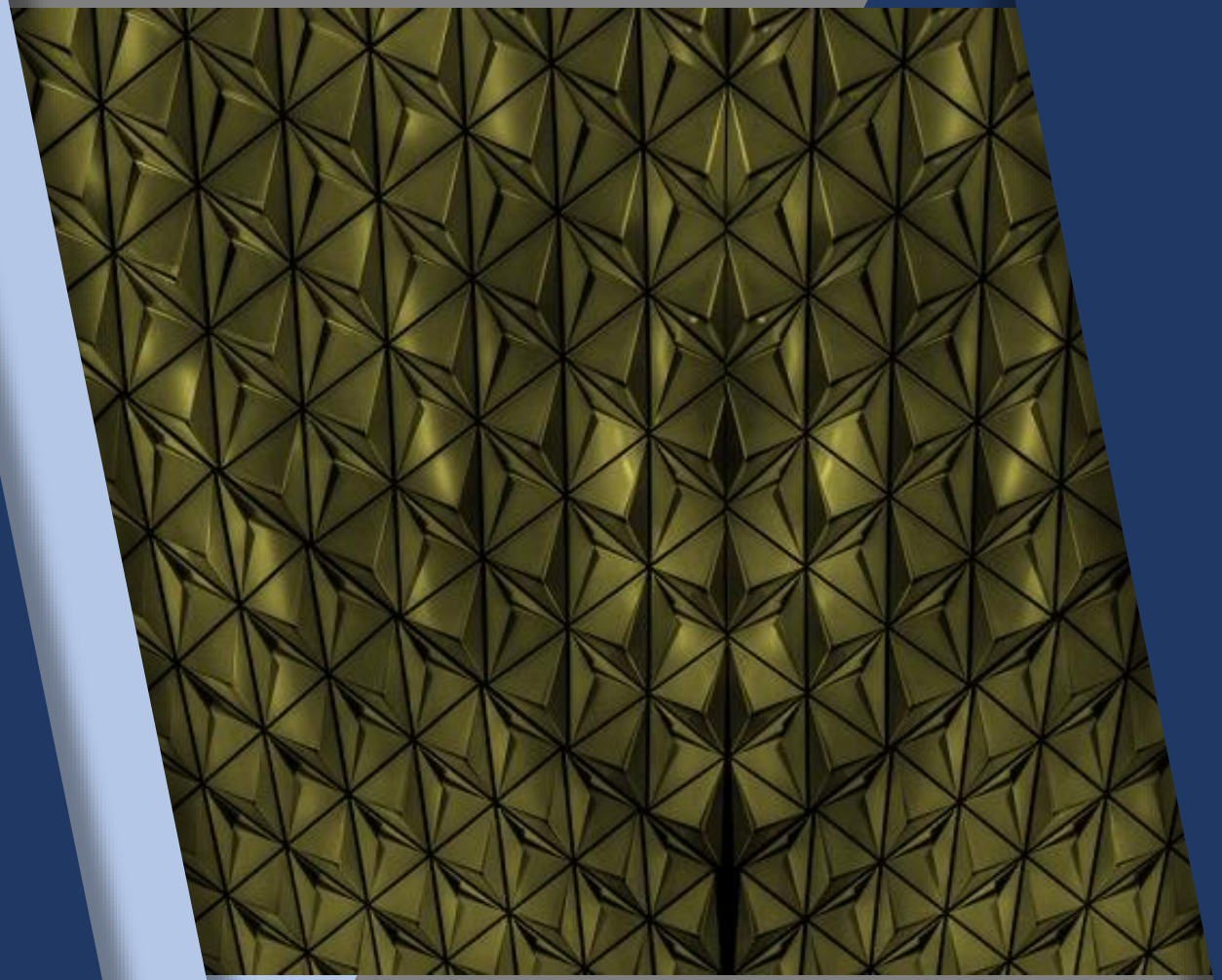
Specifikacija i modeliranje softvera

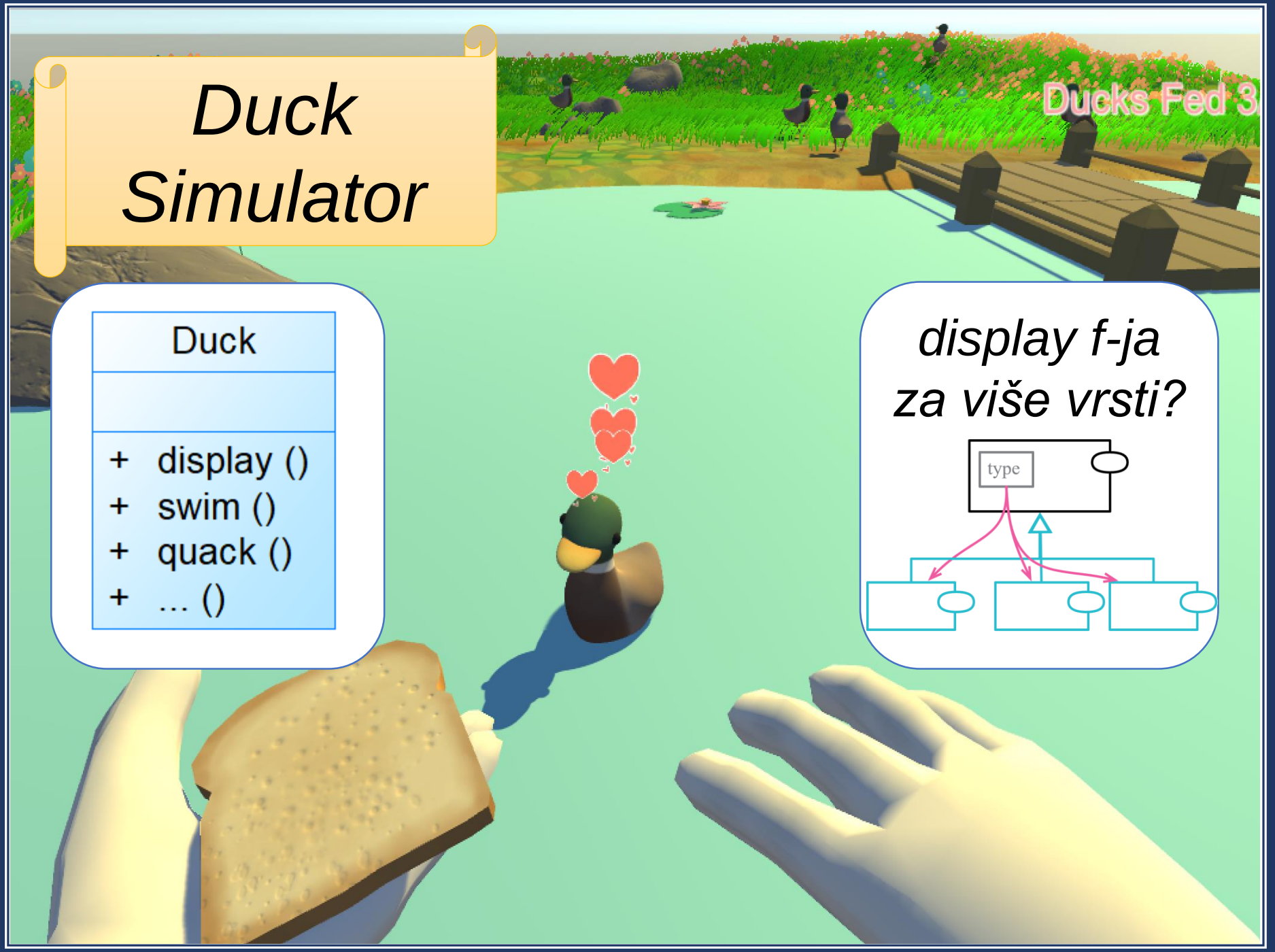


Dizajn šabloni

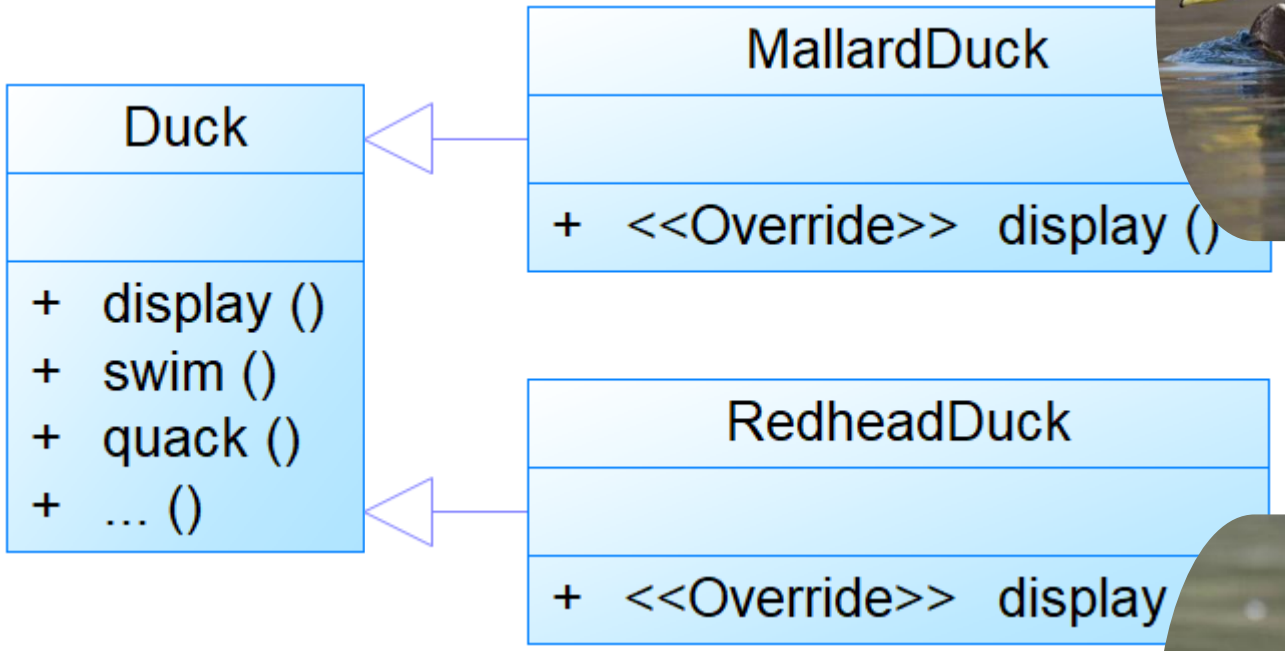
Nikola Luburić
nikola.luburic@uns.ac.rs

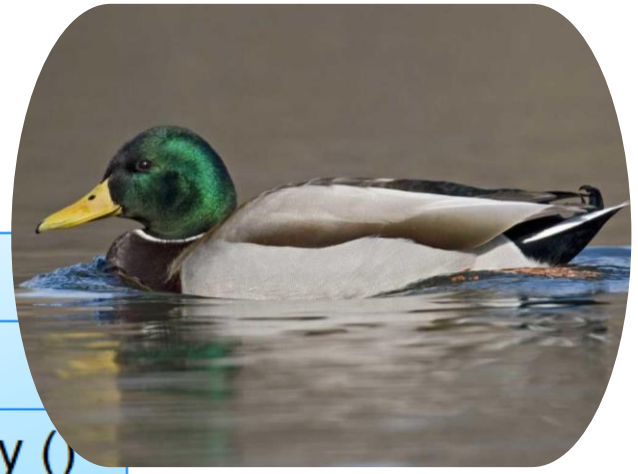
Dizajn šabloni

Dizajn šabloni



Dizajn šabloni





patke treba da lete

Dizajn šabloni

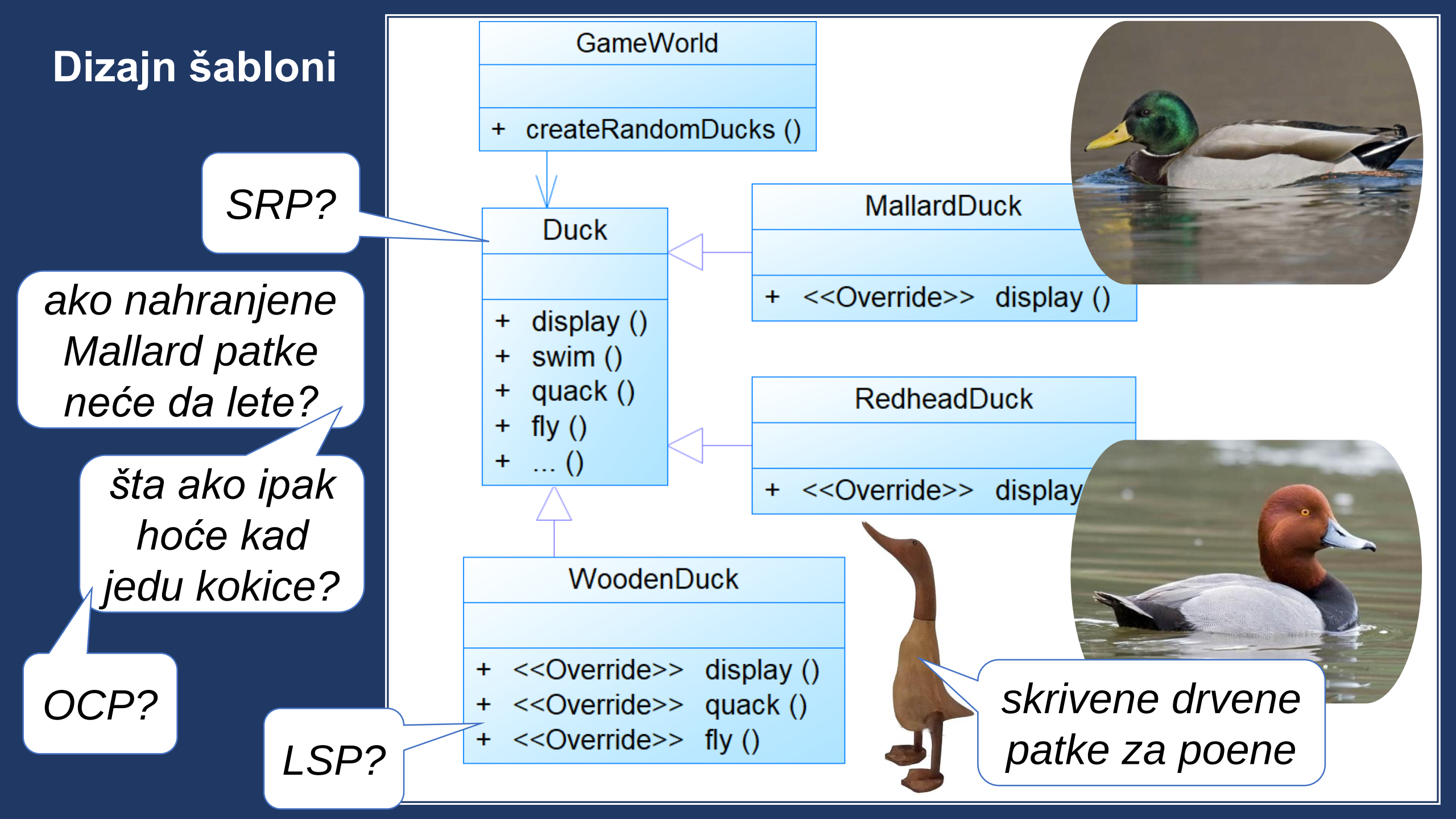
SRP?

ako nahranjene Mallard patke neće da lete?

šta ako ipak hoće kad jedu kokice?

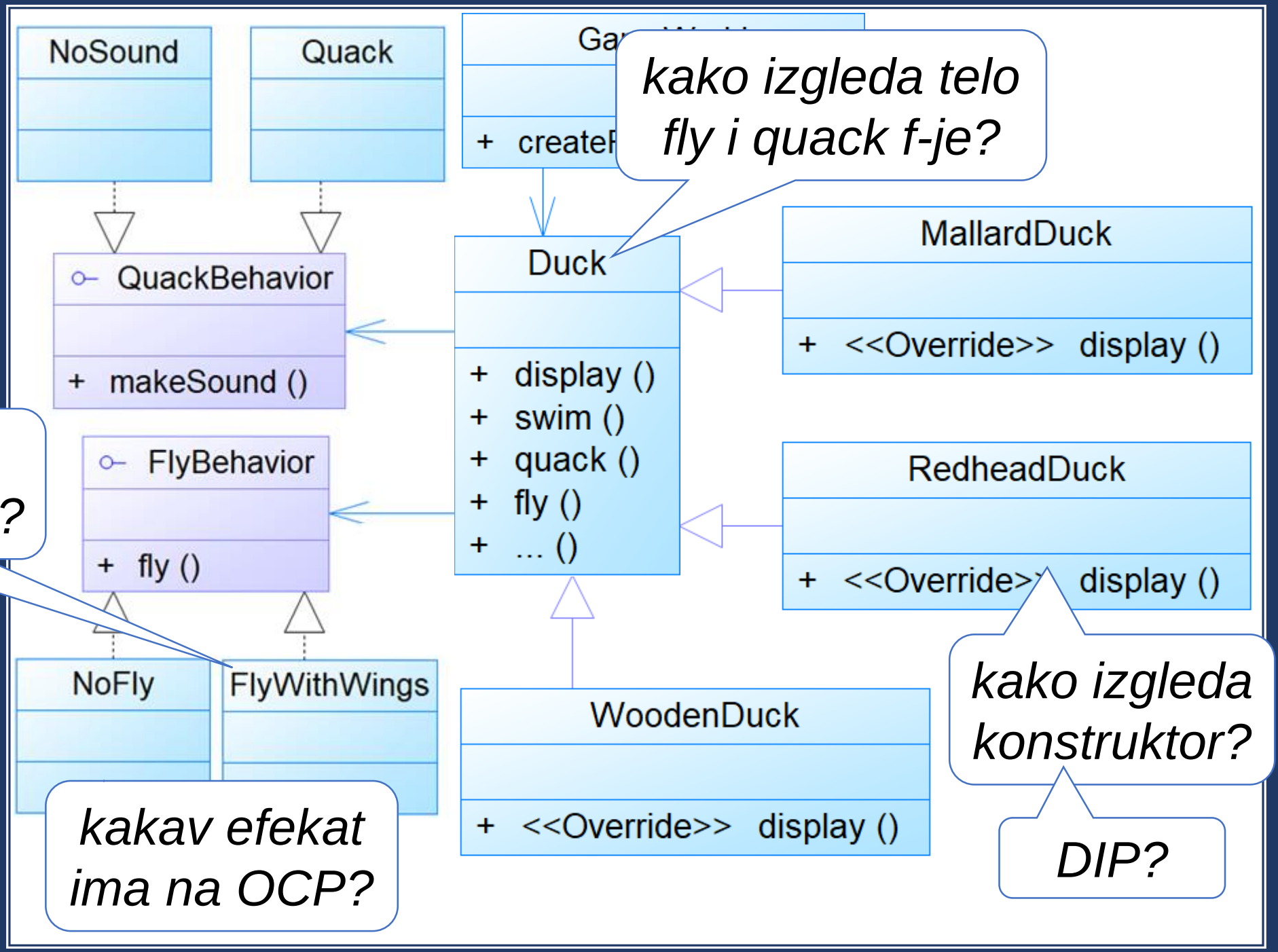
OCP?

LSP?

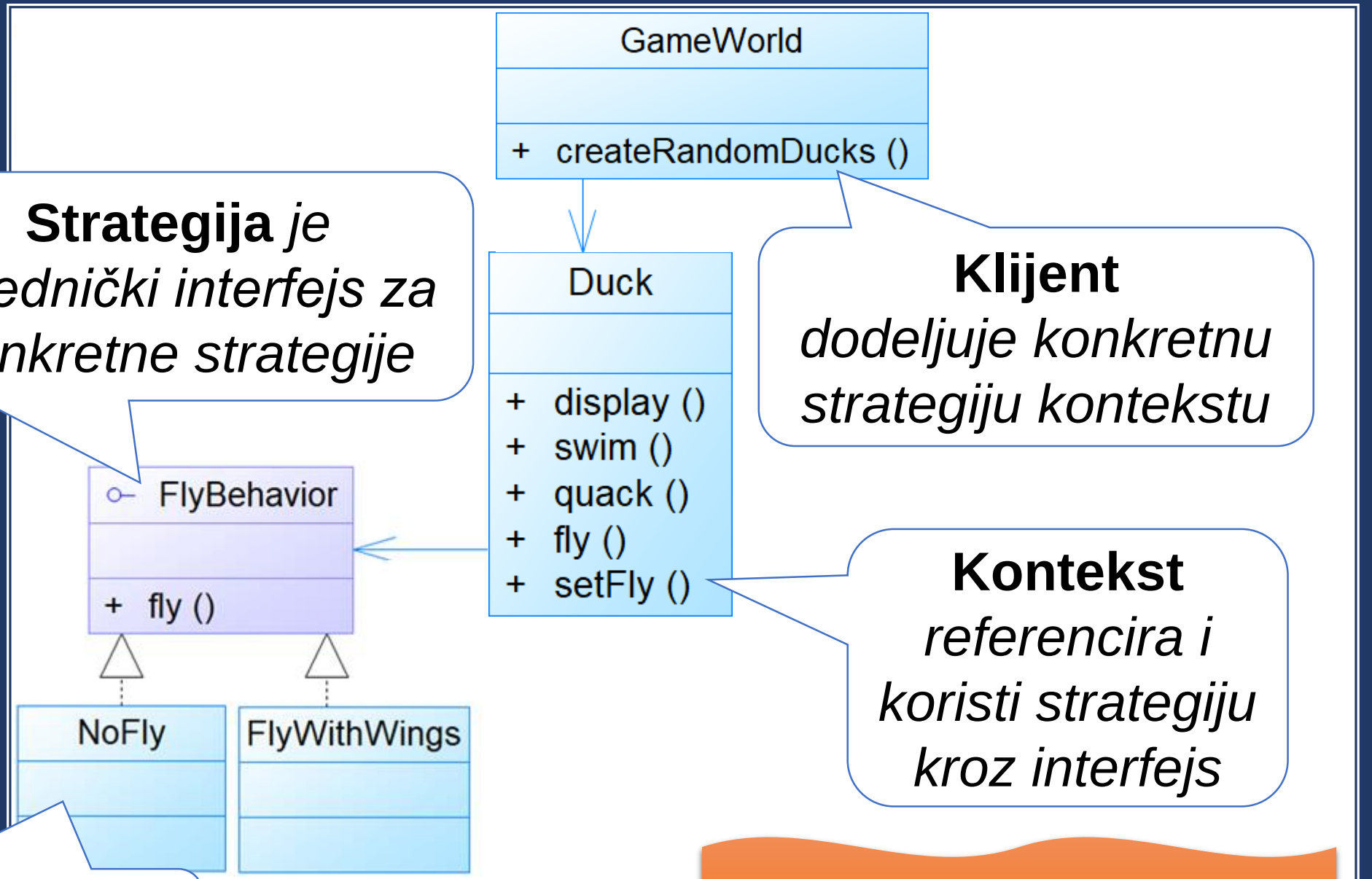


skrivene drvene patke za poene

Dizajn šabloni



Dizajn šabloni



Strategija *je zajednički interfejs za konkretne strategije*

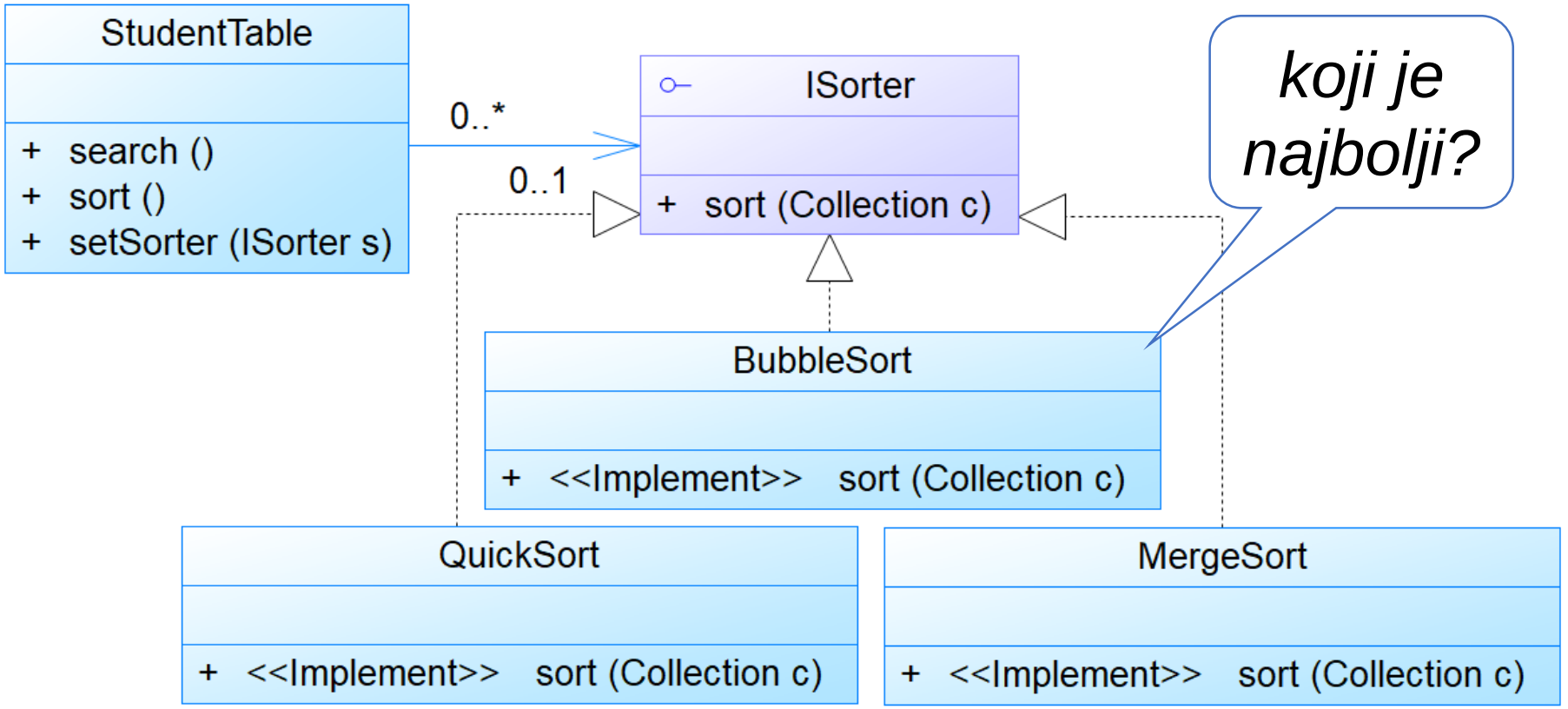
Klijent *dodeljuje konkretnu strategiju kontekstu*

Kontekst *referencira i koristi strategiju kroz interfejs*

Konkretna strategija *je implementacija algoritma*

Strategy Pattern

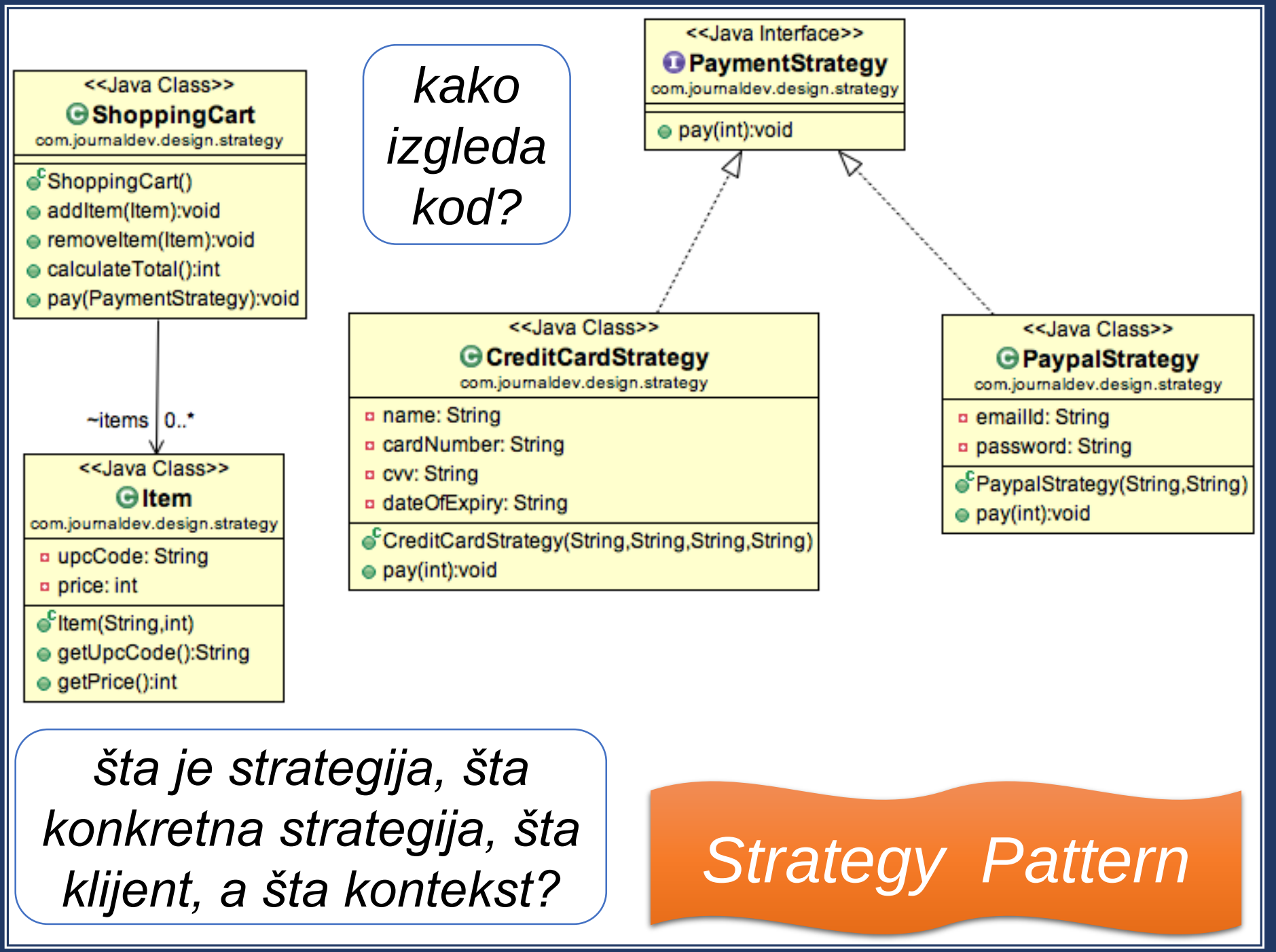
Dizajn šabloni



šta je strategija, šta konkretna strategija, šta klijent, a šta kontekst?

Strategy Pattern

Dizajn šabloni



Z13.1: Definisati klase i pseudokod za formu i njena polja, tako da se podrži validacija podataka, gde važi:

- Forma sadrži proizvoljan broj polja
- Vrednost svakog polja može da bude validirana od strane više validatora
  (npr. validiramo jedno polje da sadrži broj, da je vrednost veća od 1990 i manja od trenutne godine)
- Isti validatori se mogu primeniti na više polja

Strategy Pattern

Strategy

Definisanje familije algoritama i omogućavanje njihove zamene bez uticaja na klijenta

da li je Repository Strategy?

Dizajn šabloni

Strategy

Dizajn šabloni

Strategy

Duck Simulator

Release 1.5
Enhanced duck behavior

New features include:

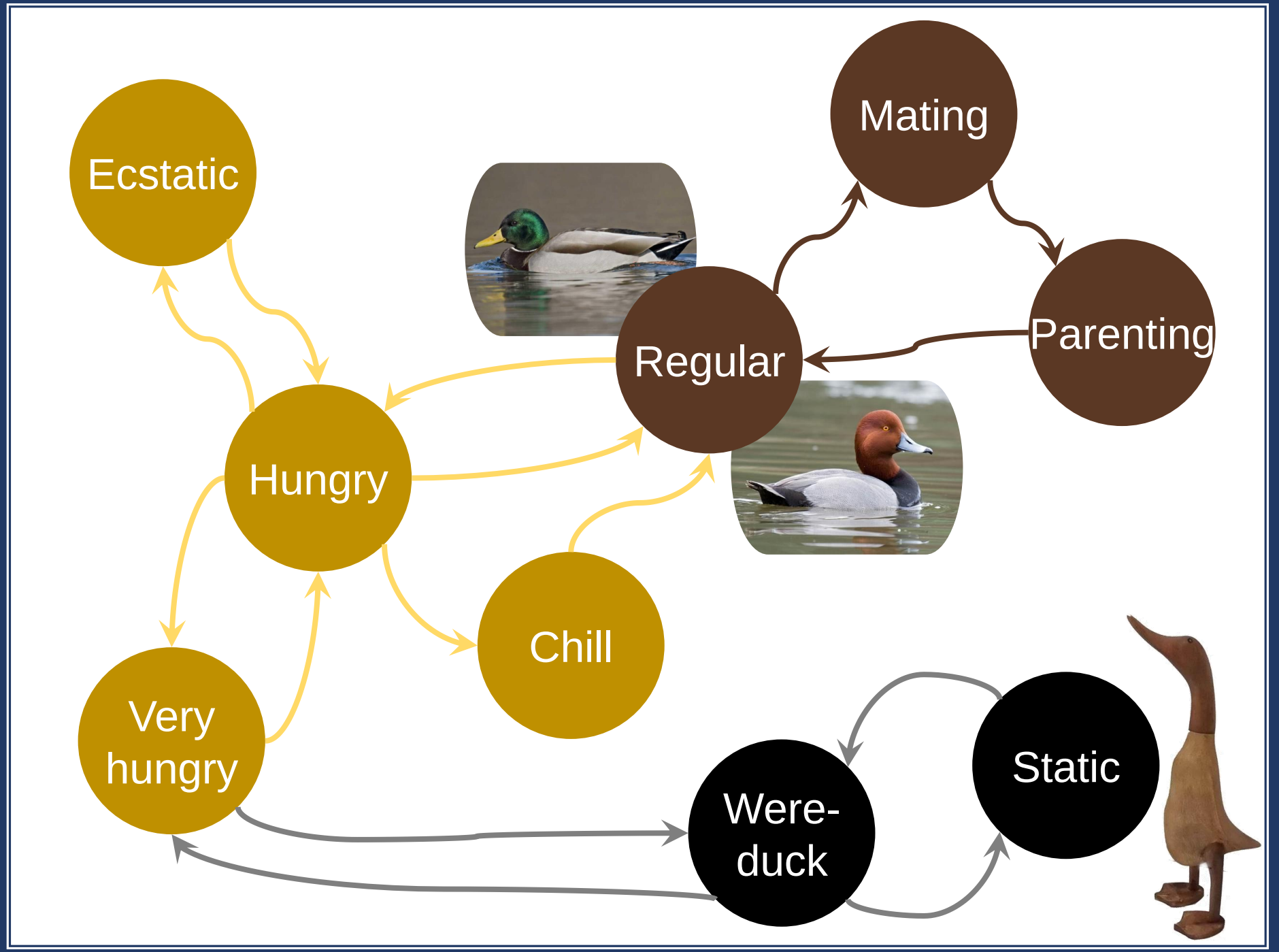
- Ducks become parents
- Ducks react differently to different food
- Ducks alter behavior based on hunger (watch out for hungry ducks!)

Bonus: Wooden ducks come alive on full moons…

Dizajn šabloni

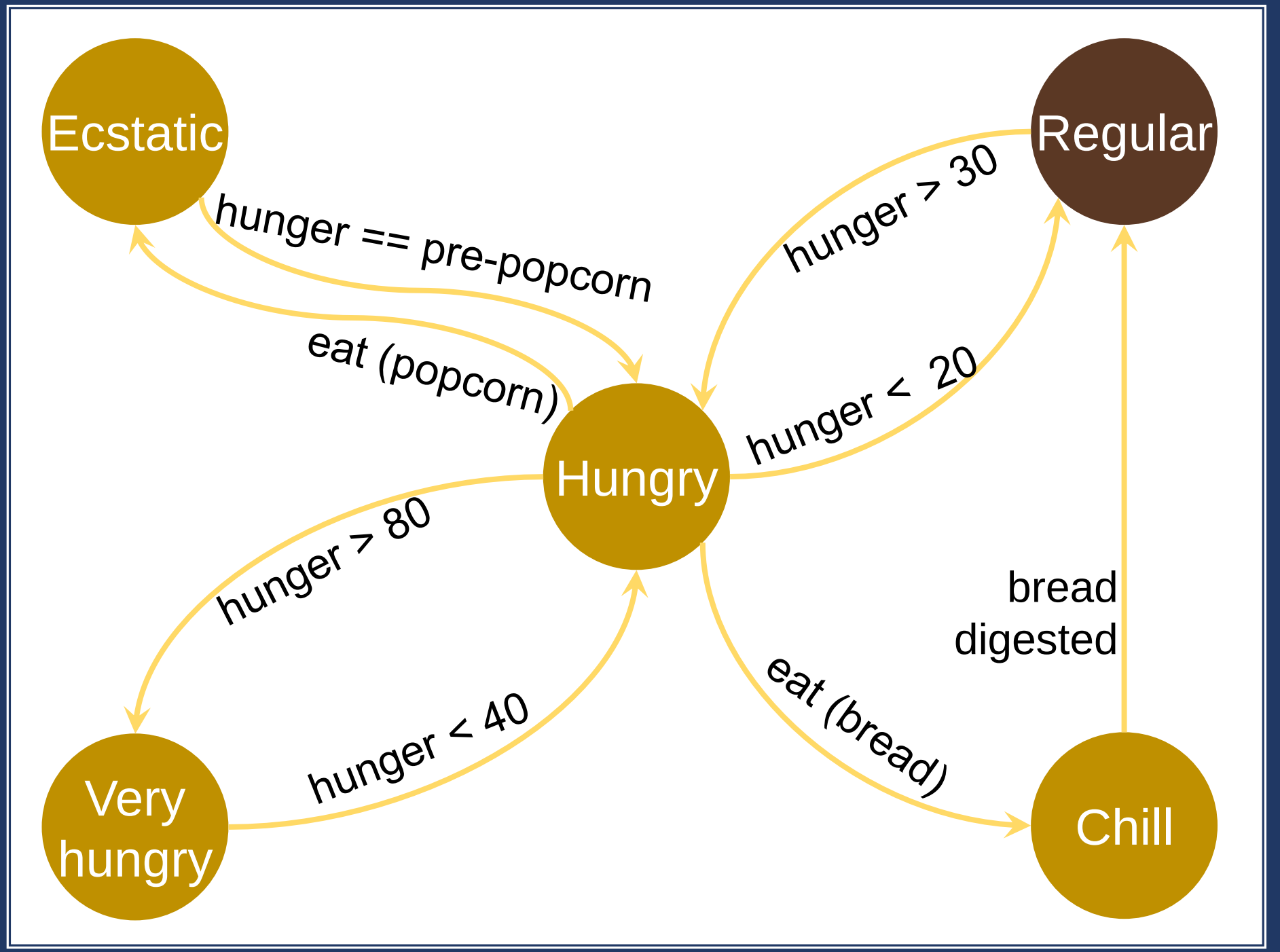
Strategy

šta vidimo?



Dizajn šabloni

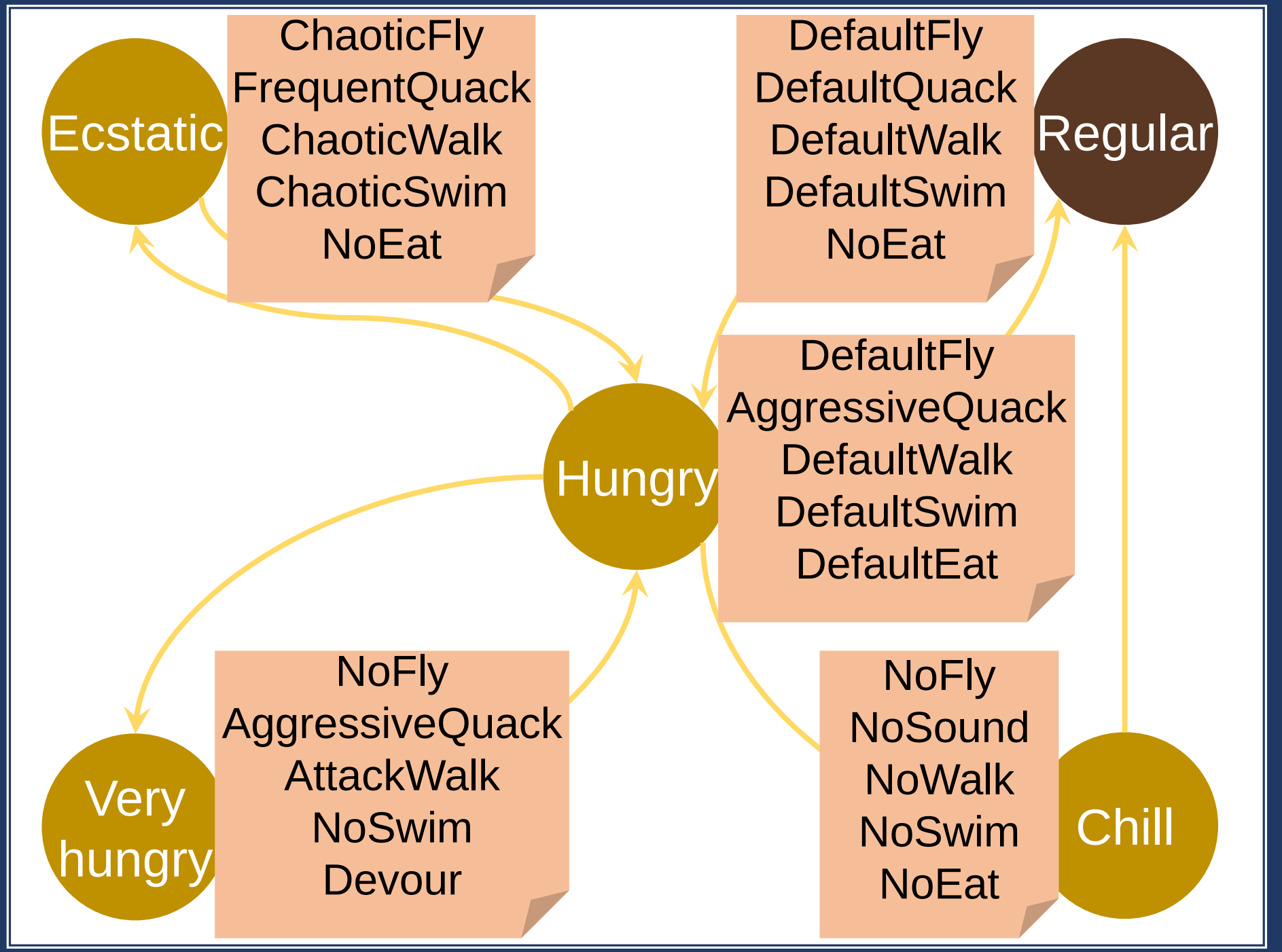
Strategy



Dizajn šabloni

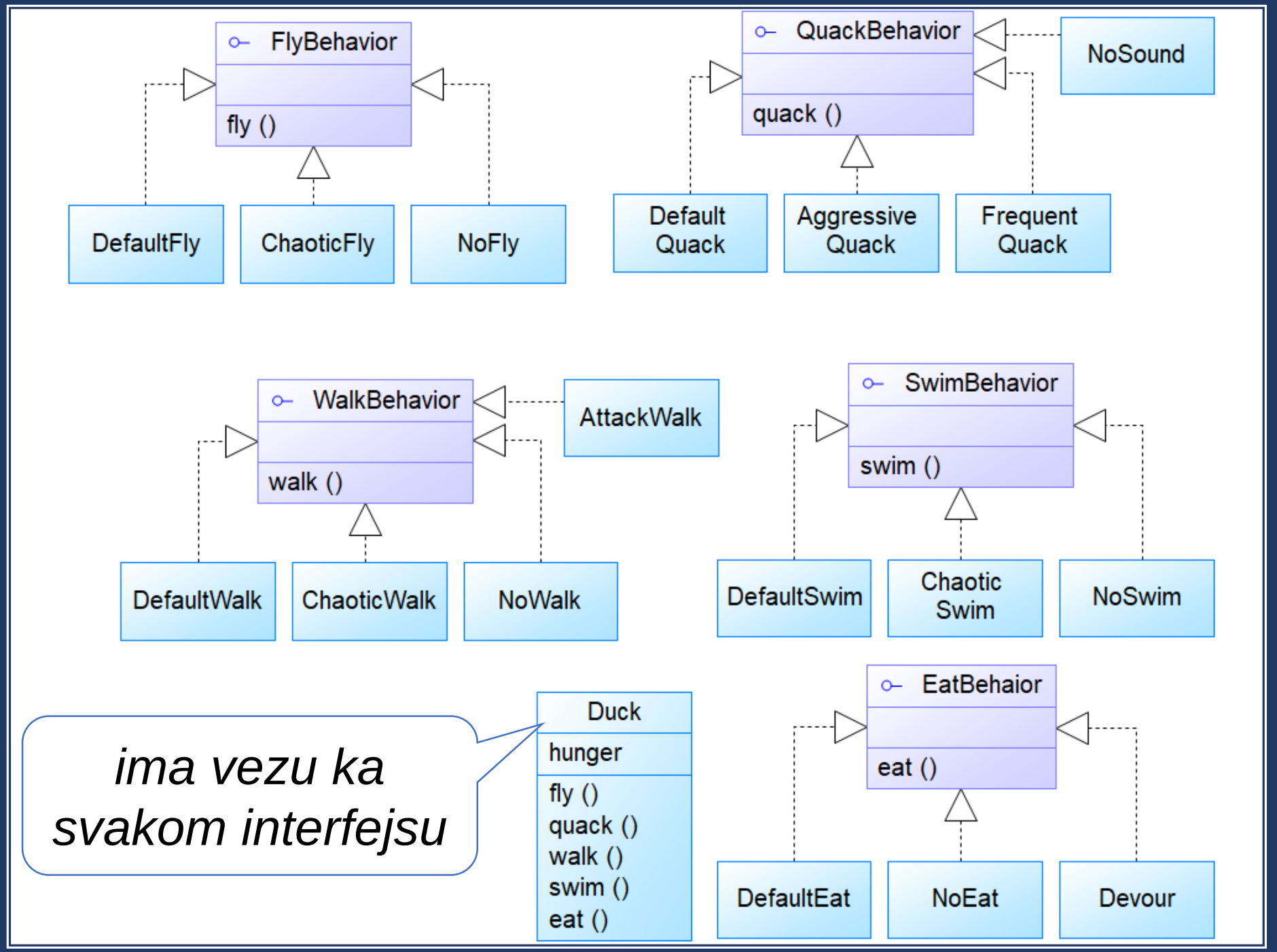
Strategy

kako izgleda dijagram klase?



Dizajn šabloni

Strategy



Dizajn šabloni

Strategy

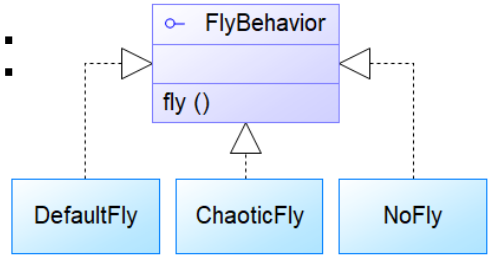
šta je problem?

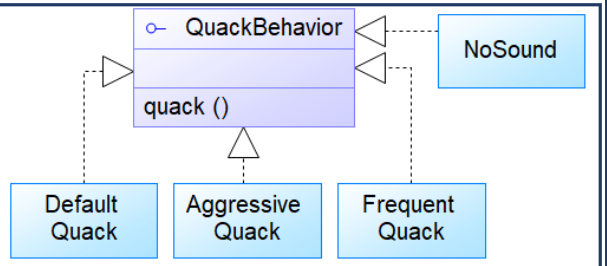
Duck *brine o sebi i promeni skupa strategija spram stanja*

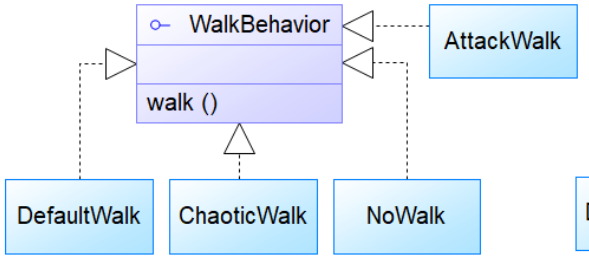
šta ako ubacimo druga stanja?

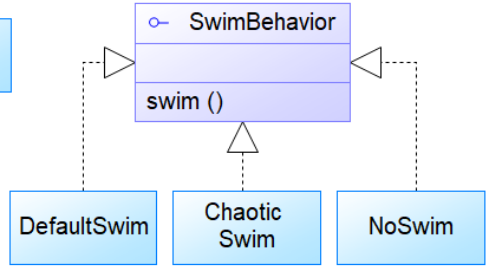
Napiši pseudokod za:

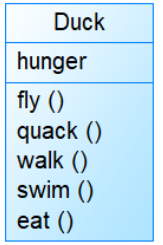
- Prelazak patke iz stanja *Hungry* u *Ecstatic* i obratno
- *Fly*, *walk* i *swim* povećava *hunger*

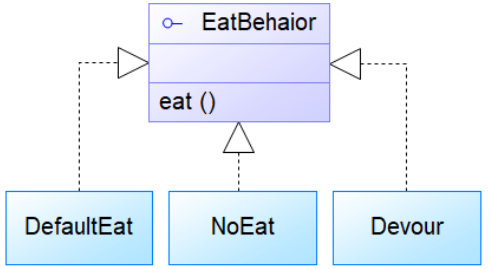




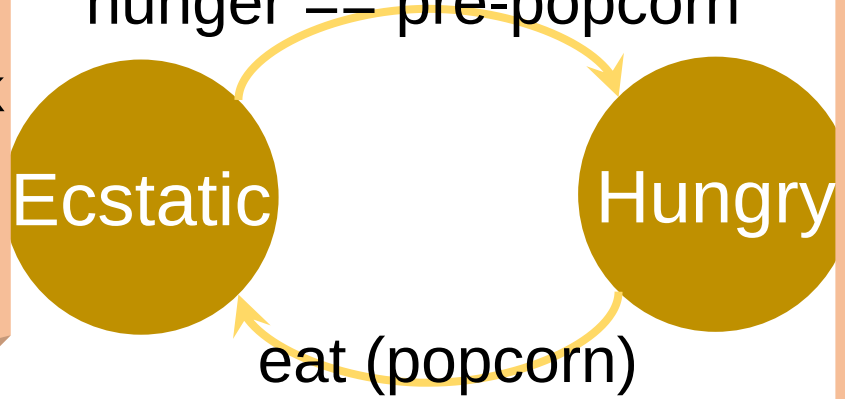








ChaoticFly
FrequentQuack
ChaoticWalk
ChaoticSwim
NoEat



DefaultFly
AggressiveQuack
DefaultWalk
DefaultSwim
DefaultEat

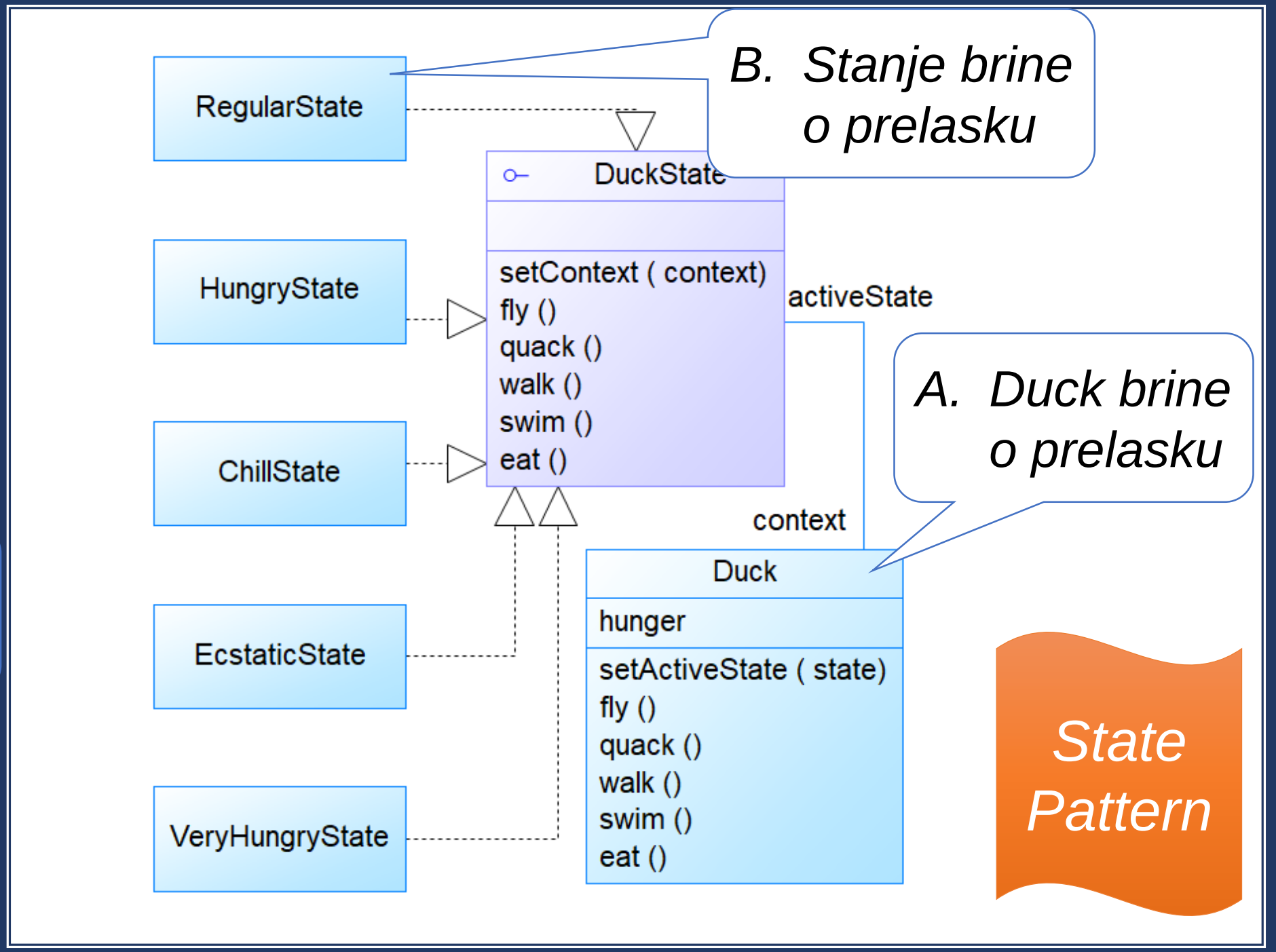
Dizajn šabloni

Strategy

prednosti i mane?

kako prelazimo iz stanja u stanje?

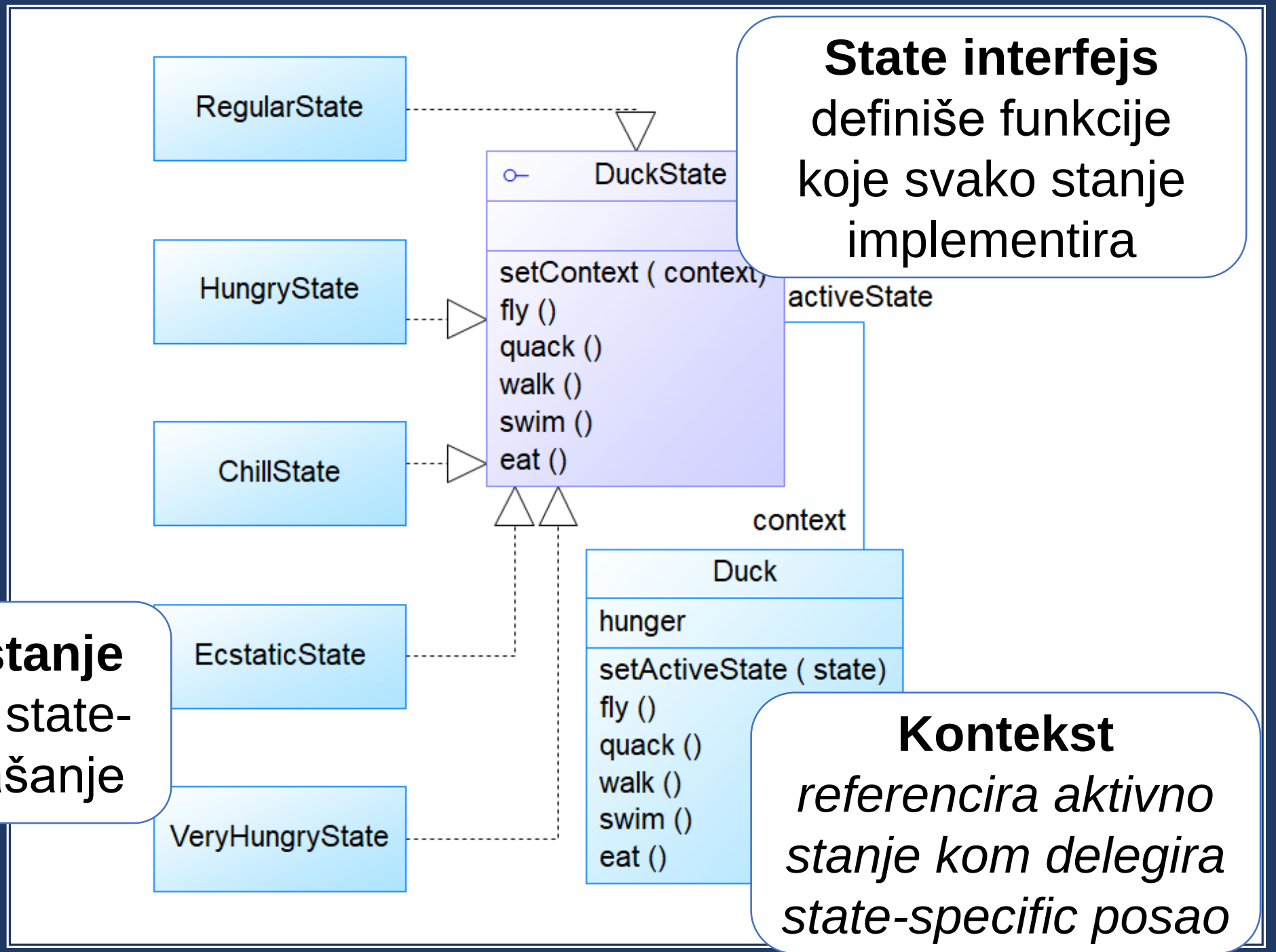
kako izgleda kod?



Dizajn šabloni

Strategy

State



State interfejs
definiše funkcije koje svako stanje implementira

Konkretno stanje
implementira state-specific ponašanje

Kontekst
referencira aktivno stanje kom delegira state-specific posao

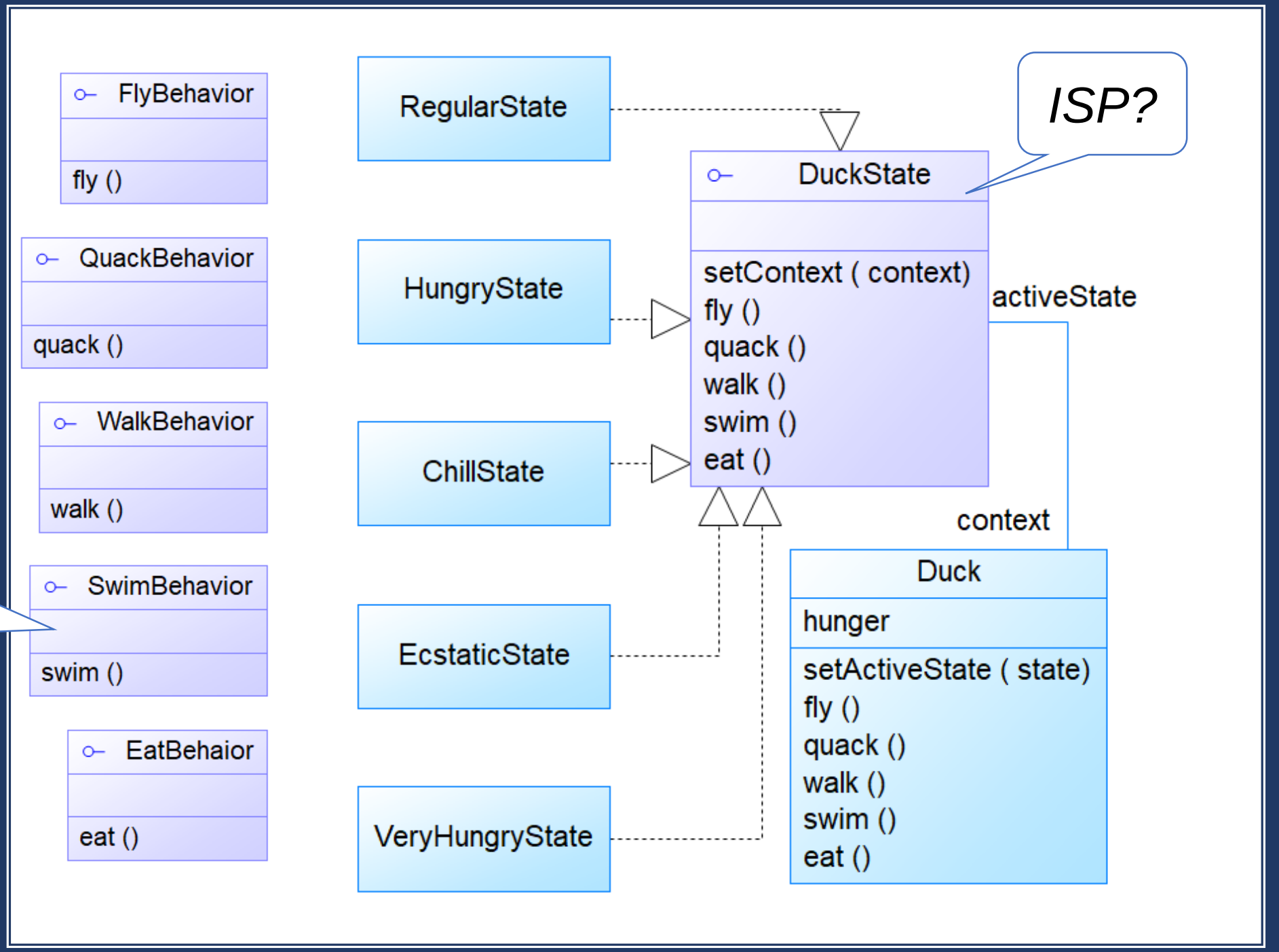
Dizajn šabloni

Strategy

State

možemo kombinovati sa strategy

prednosti i mane?



Dizajn šabloni

Strategy

State

Definisanje familije stanja i omogućavanje izmene celokupnog ponašanja objekta

Dizajn šabloni

Strategy

State

History

Data Feed

Current

Forecast

Weather App

Dizajn šabloni

Strategy

State

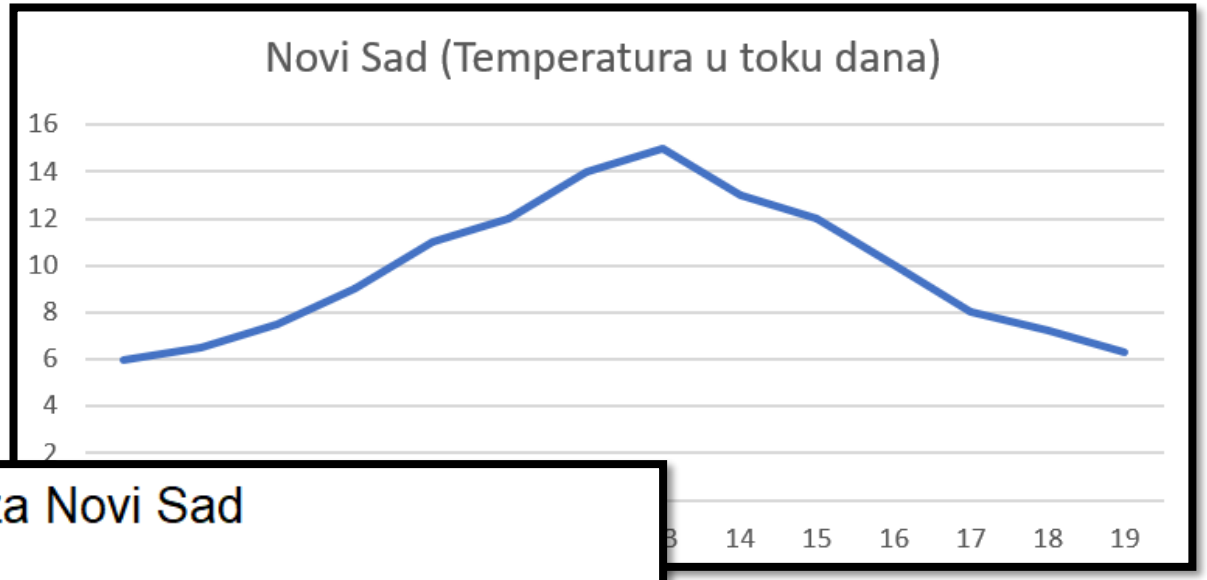
šta kada stigne novi podatak?

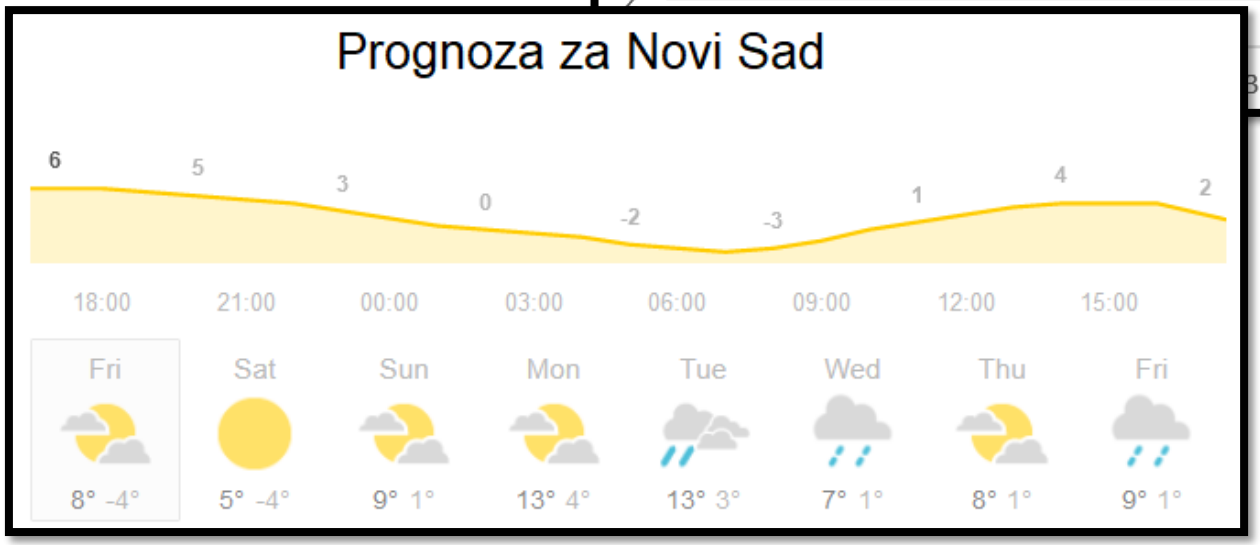
Data Feed

Trenutne vrednosti meteroloških merenja			
Lokacija	Temp. (ºC)	Vetar (km/h)	Vlažnost (%)
Novi Sad	6,3	10,5	73

Current

History





Forecast

Dizajn šabloni

Strategy

State

šta vidimo?

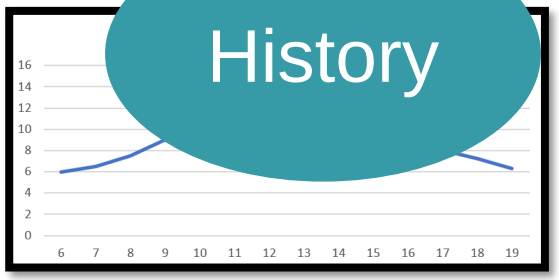
CurrentWeather
- temperature : double - windSpeed : double - humidity : int
+ updateDisplay ()

Current

Trenutne vredno merenja			
Lokacija	Temp. (ºC)	Vetar (km/h)	Vlažnost (%)
Novi Sad	6,3	10,5	73

History
- historicalData : double[]
+ updateDisplay ()

History



DataFeed
- lastTemperature : double - lastWindSpeed : double - lastHumidity : int
+ getTemperature () + getWindSpeed () + getHumidity () + pullNewData ()

Data Feed

Forecast
- weatherData : double[][]
+ updateDisplay () + calculateForecast ()

ForecastStrategy

Forecast

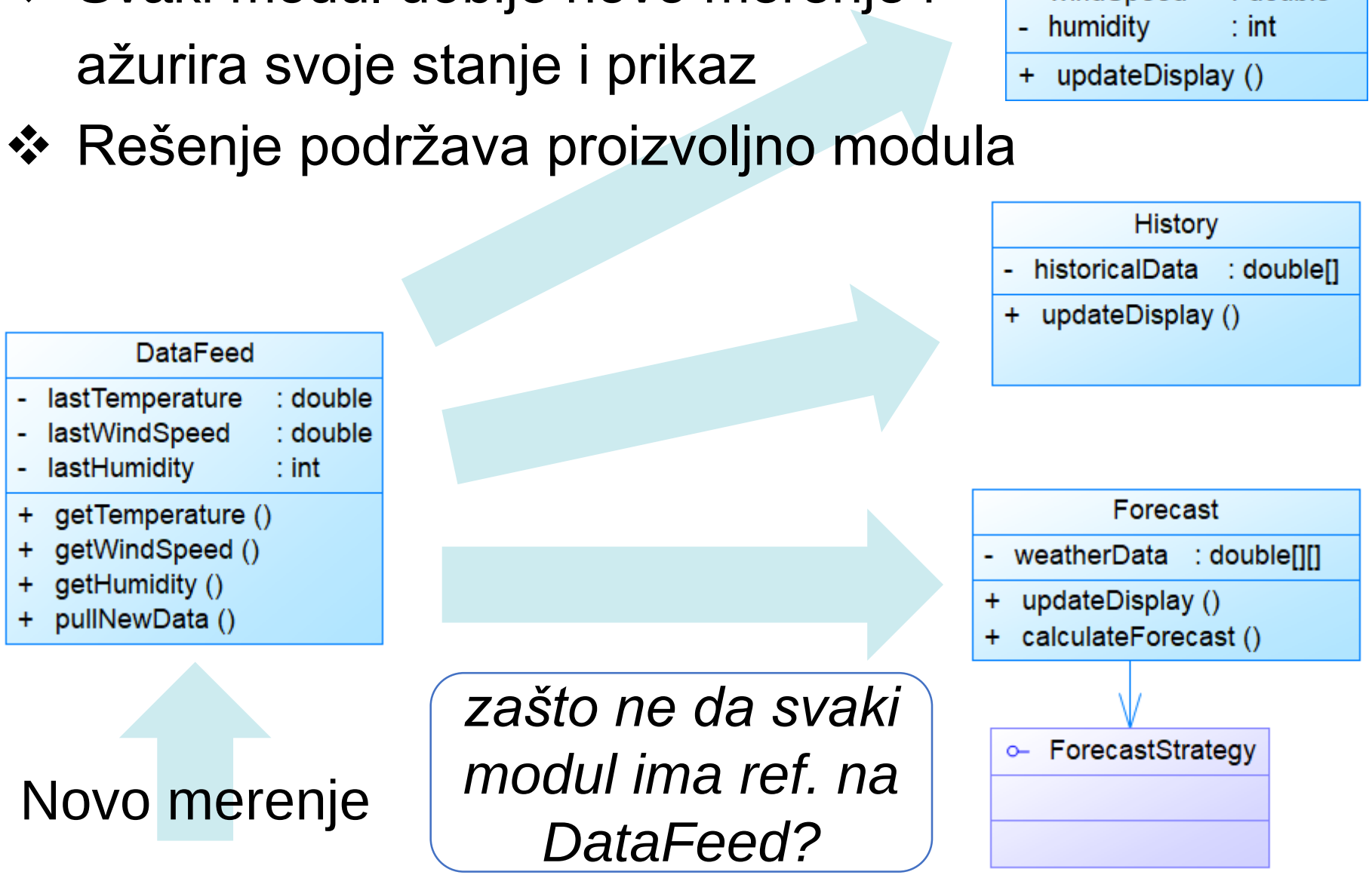
Dizajn šabloni

Strategy

State

Proširi dijagram i napiši kod da:

- Svaki modul dobije novo merenje i ažurira svoje stanje i prikaz
- Rešenje podržava proizvoljno modula



Dizajn šabloni

Strategy

Proširi dijagram i napiši kod da:

- Svaki modul dobije novo merenje i ažurira nje i prikaz
- Rešenj va proizvoljno modula

gledaš više feed-ova?

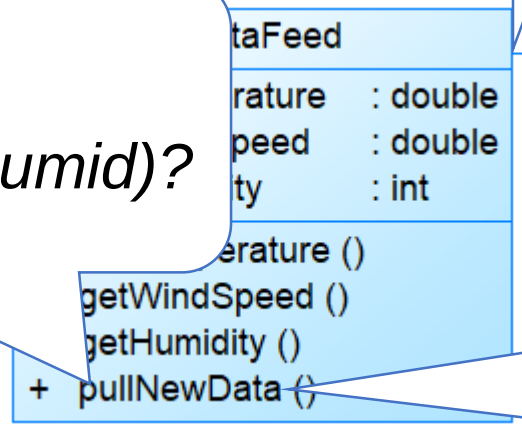
update(this)?
update(temp, speed, humid)?

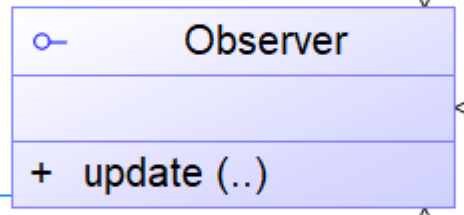
proširivost vs enkapsulacija

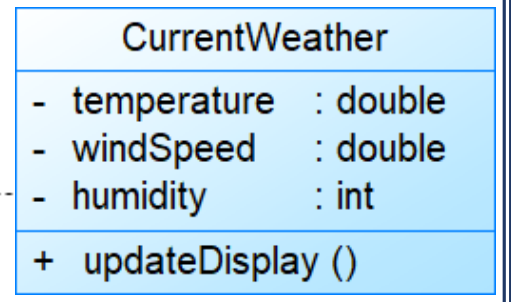
kako izgleda kod?

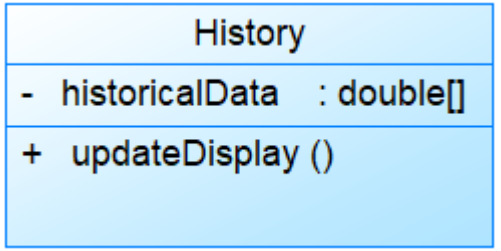
kako izgleda kod?

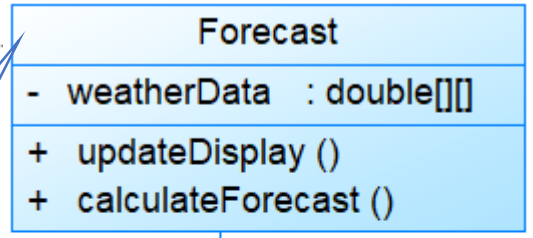
kako izgleda kod?

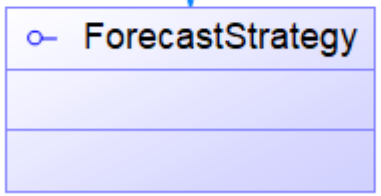












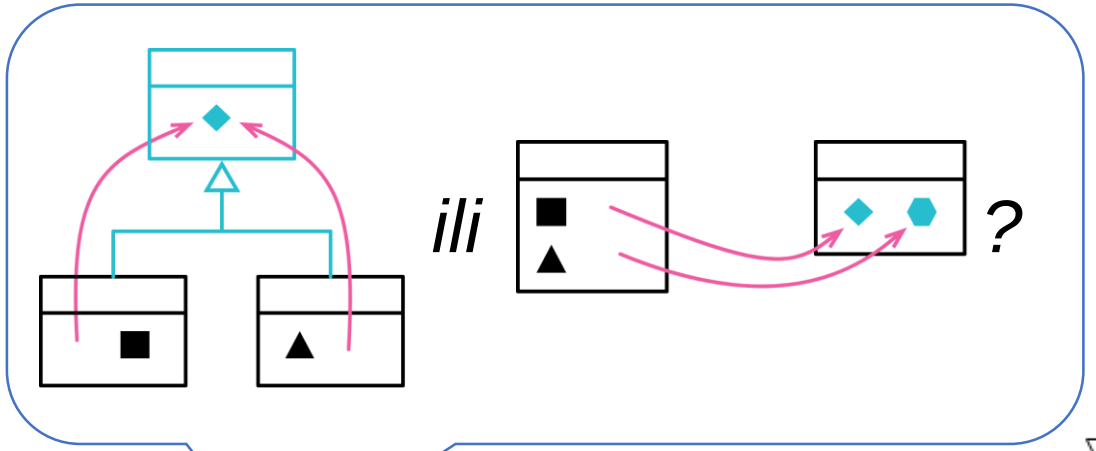
Novo merenje

Dizajn šabloni

Strategy

State

kohezija?



CurrentWeather
- temperature : double - windSpeed : double - humidity : int
+ updateDisplay ()

DataFeed
- lastTemperature : double - lastWindSpeed : double - lastHumidity : int
+ getTemperature () + getWindSpeed () + getHumidity () + pullNewData () + addObserver (Observer o) + removeObserver (Observer o)

Observer
+ update (..)

History
- historicalData : double[]
+ updateDisplay ()

Forecast
- weatherData : double[][]
+ updateDisplay () + calculateForecast ()

ForecastStrategy

Novo merenje

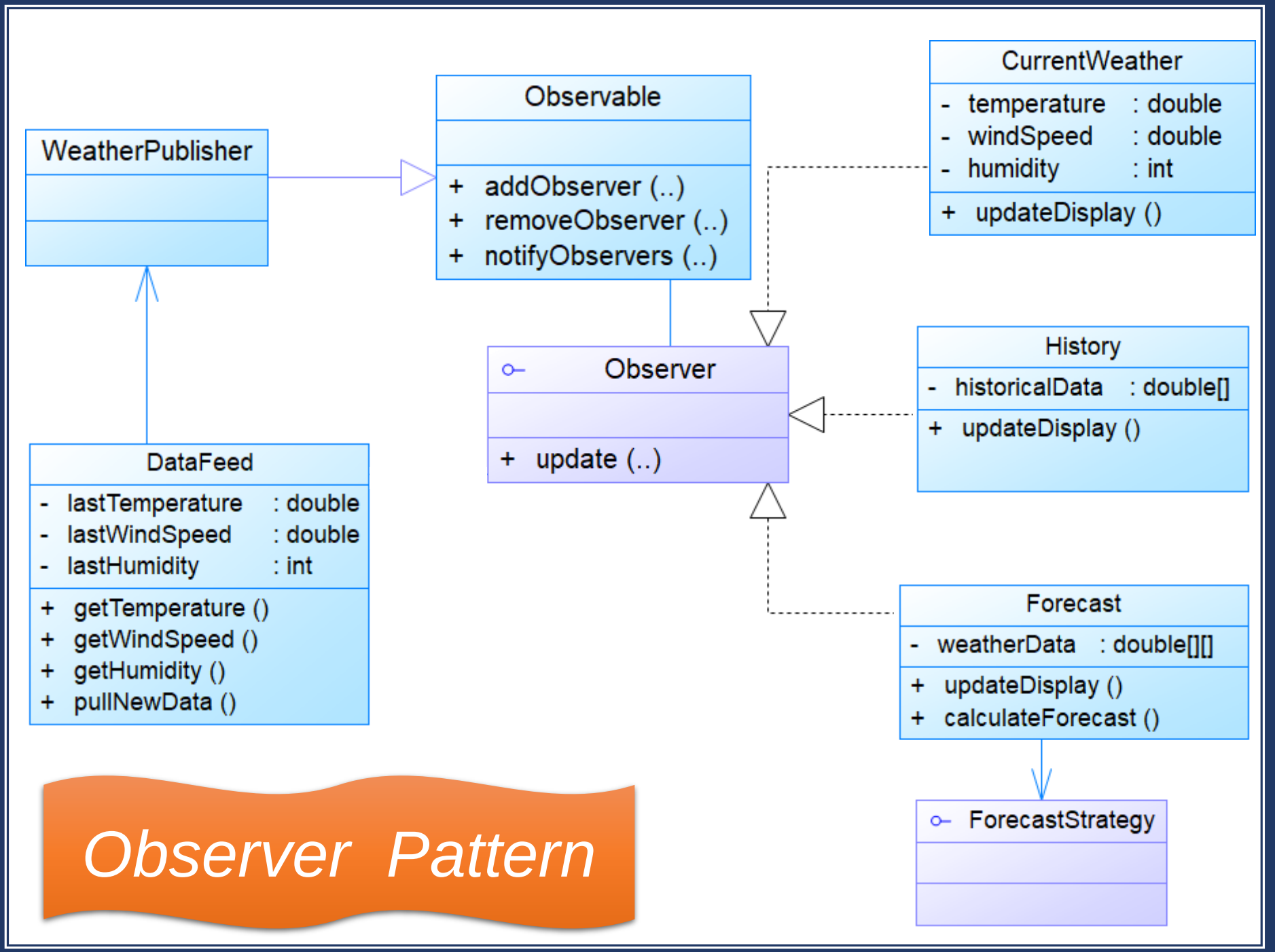
Dizajn šabloni

Strategy

Šta ako veoma sitne izmene stižu često, a crtanje je skupo?

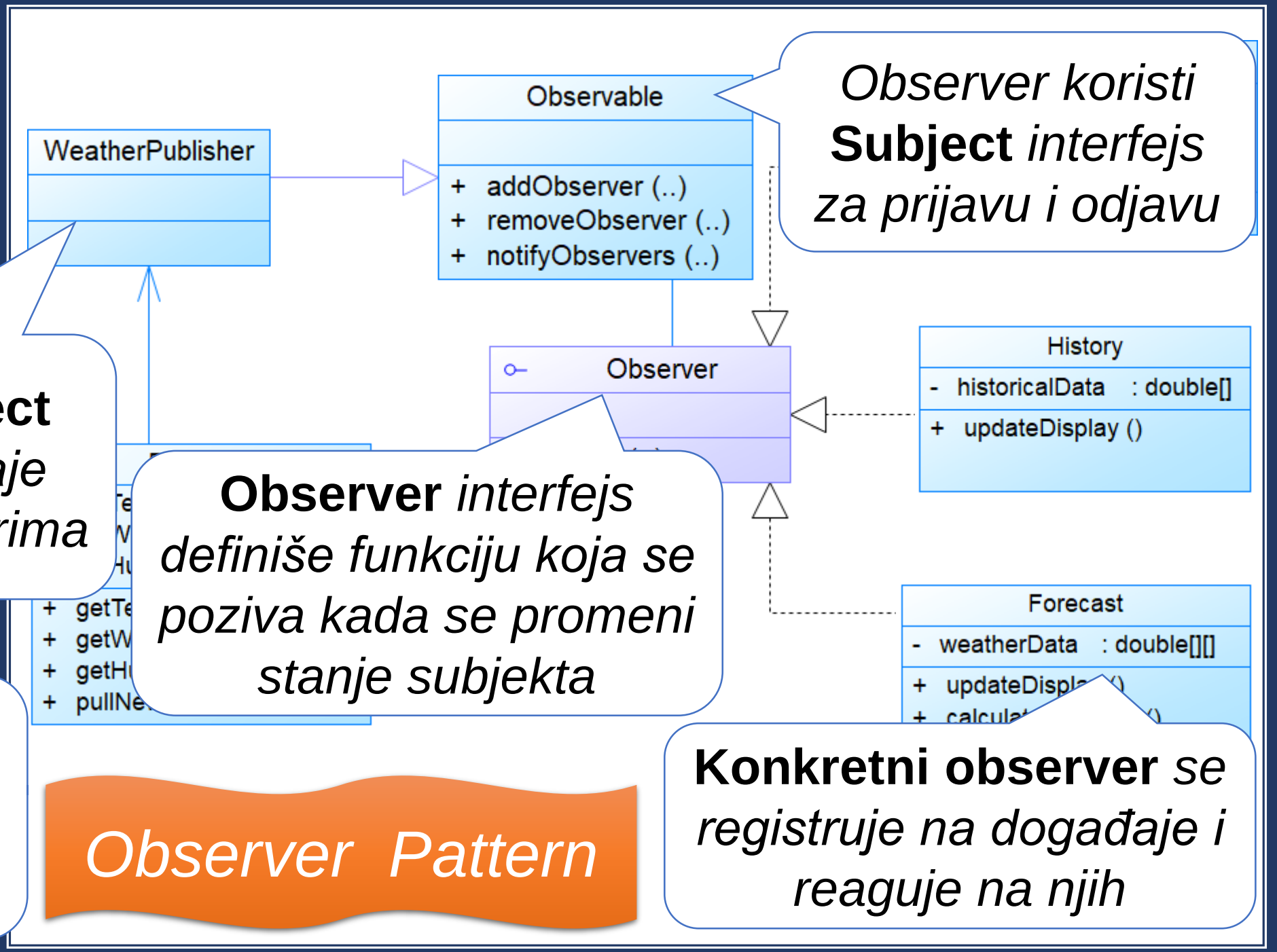
Šta ako ima puno observera koji su zainteresovani za izmenu dela stanja?

Z13.2: Koncipirati dizajn i kod rešenja



Dizajn šabloni

Strategy



Konkretni subject *obrađuje događaje koje šalje observerima*

Observer koristi **Subject** *interfejs za prijavu i odjavu*

Observer *interfejs definiše funkciju koja se poziva kada se promeni stanje subjekta*

Z13.3: *Istražiti razliku između observer i PubSub mehanizma*

Observer Pattern

Konkretni observer *se registruje na događaje i reaguje na njih*

Dizajn šabloni

Strategy

State

Observer

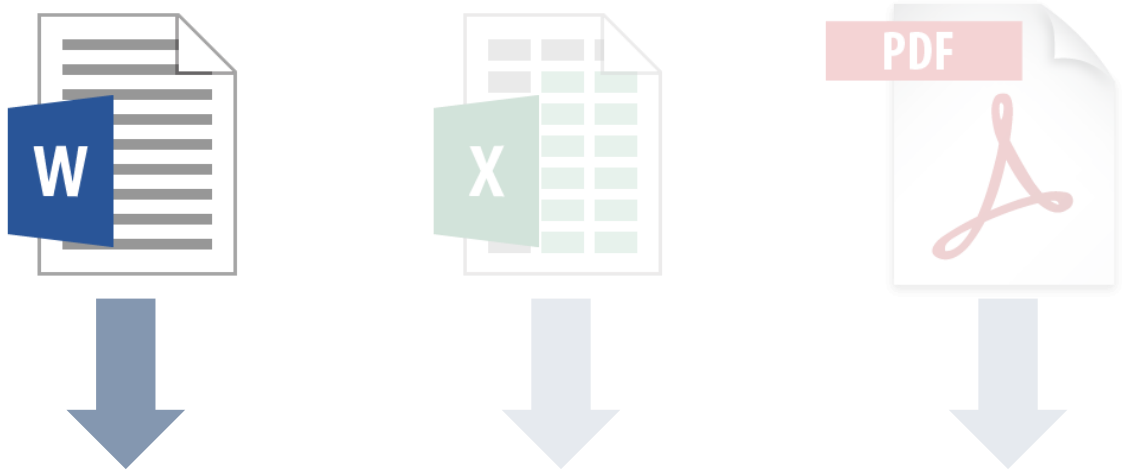
Definisanje odnosa među objektima, gde jedan održava listu povezanih objekata koje obaveštava o svojim izmenama

Dizajn šabloni

- *Strategy*
- *State*
- *Observer*

Dizajn šabloni

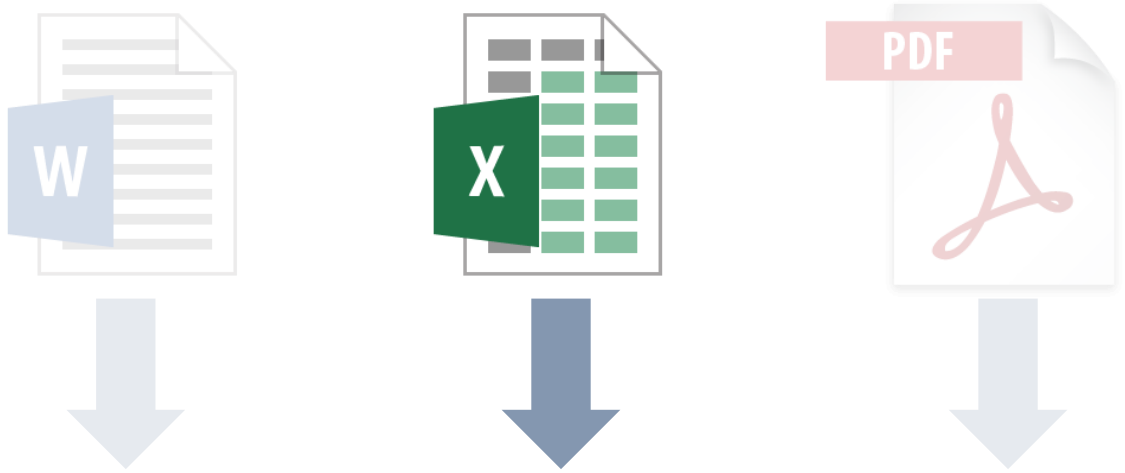
- *Strategy*
- *State*
- *Observer*




```
processWord(path) {
  file = openFile(path);
  rawData = extractDocData(file);
  data = parseDocData(rawData);
  structData = validateData(data);
  persistData(structData);
  closeFile(file);
}
```

Dizajn šabloni

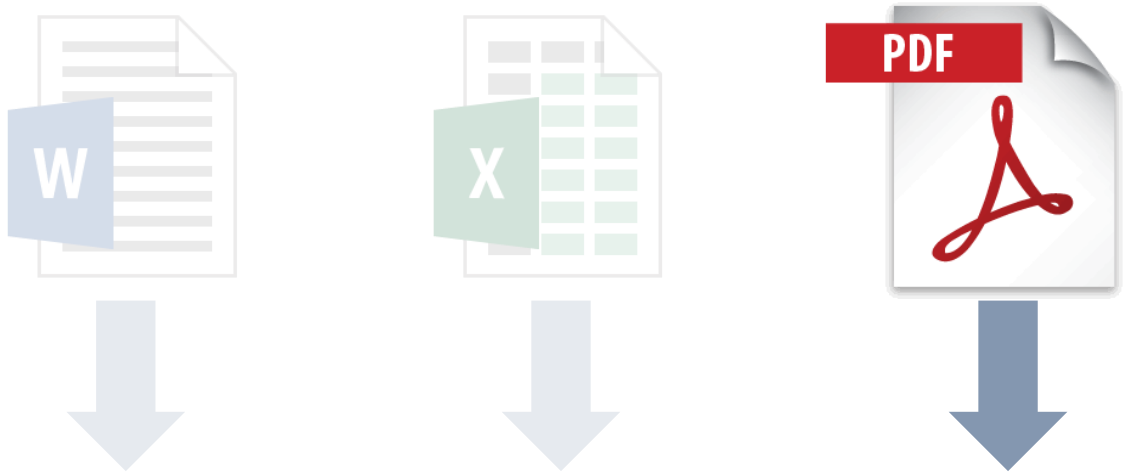
- *Strategy*
- *State*
- *Observer*




```
processExcel(path) {
  file = openFile(path);
  rawData = extractXlsData(file);
  data = parseXlsData(rawData);
  structData = validateData(data);
  persistData(structData);
  closeFile(file);
}
```

Dizajn šabloni

- *Strategy*
- *State*
- *Observer*



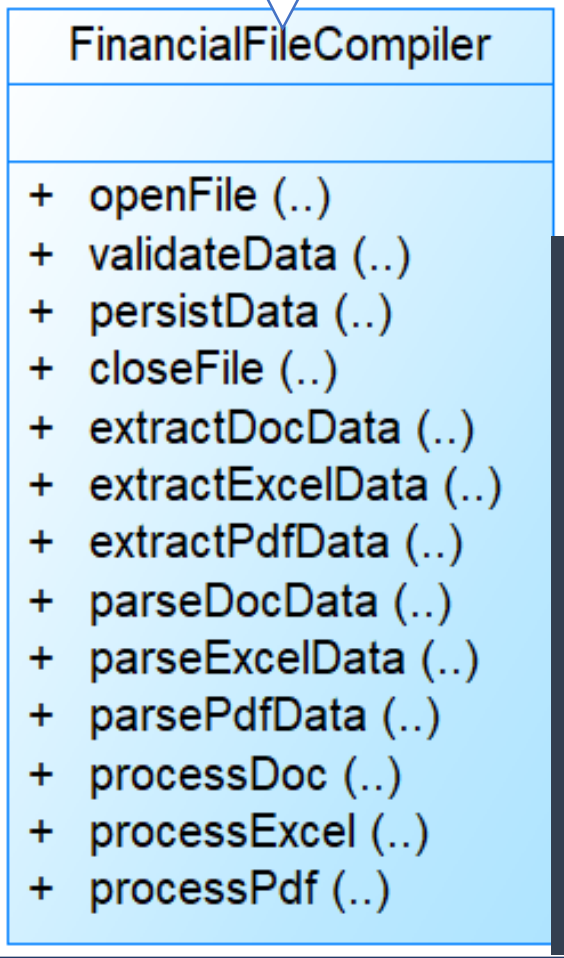

```
processPdf(path) {
  file = openFile(path);
  rawData = extractPdfData(file);
  data = parsePdfData(rawData);
  structData = validateData(data);
  persistData(structData);
  closeFile(file);
}
```

Dizajn šabloni

- Strategy
- State
- Observer

šta sve može biti razlog izmene klase?

kako izgleda klasa koja sadrži ovu funkciju?




```
processPdf(path) {
  file = openFile(path);
  rawData = extractPdfData(file);
  data = parsePdfData(rawData);
  structData = validateData(data);
  persistData(structData);
  closeFile(file);
}
```

Dizajn šabloni

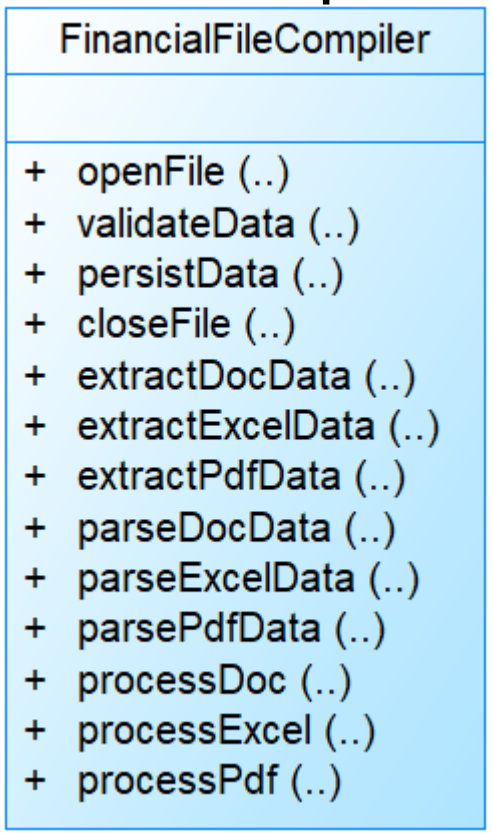
Strategy

State

Observer

Definiši rešenje koje razdvaja promenljiv od nepromenljivog koda, tako da je:

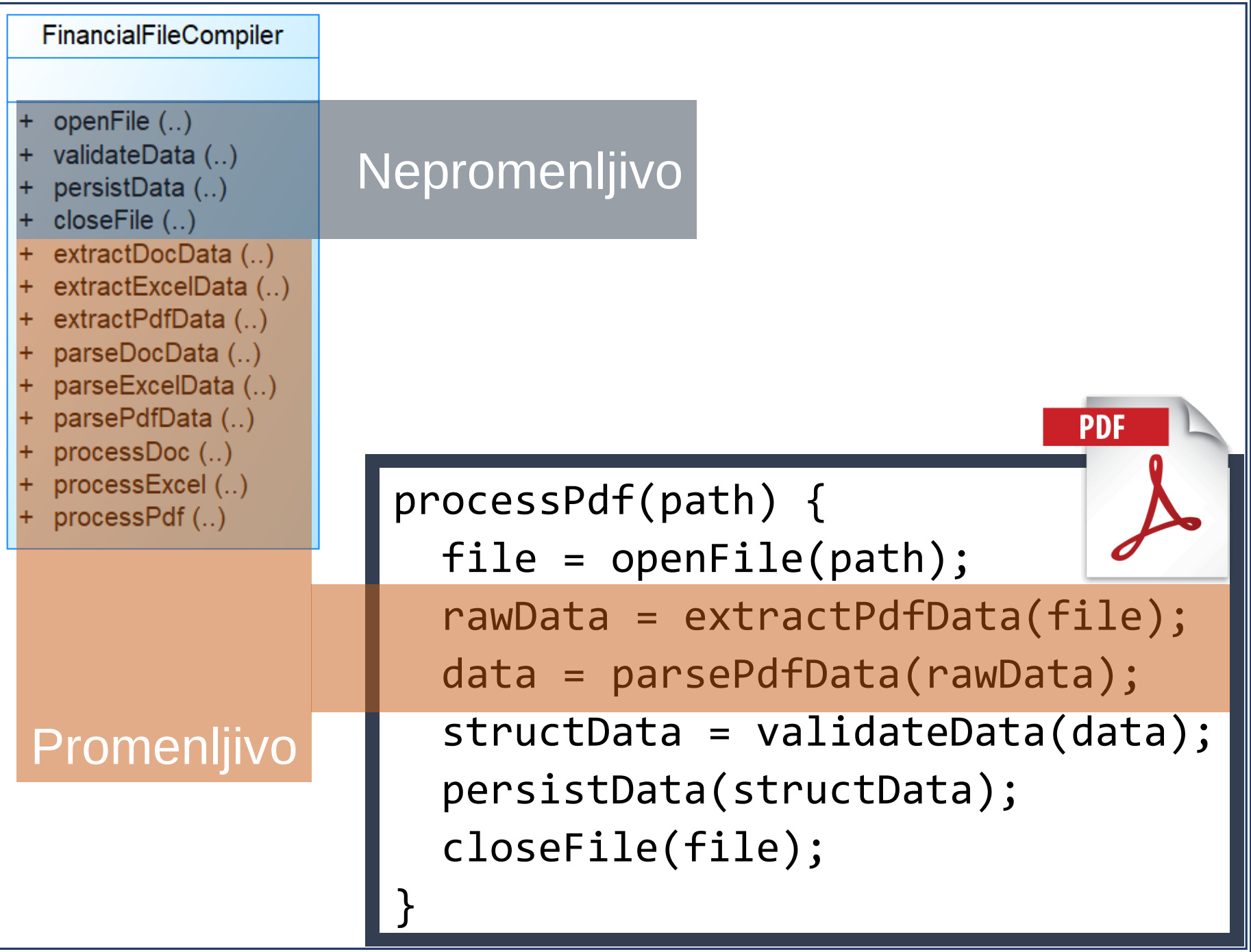
- Jednostavno podržati novi tip datoteke (npr. CSV)
- Jednostavno dodati poziv `backupFile` funkcije tokom procesiranja svih datoteka




```
processPdf(path) {
  file = openFile(path);
  rawData = extractPdfData(file);
  data = parsePdfData(rawData);
  structData = validateData(data);
  persistData(structData);
  closeFile(file);
}
```

Dizajn šabloni

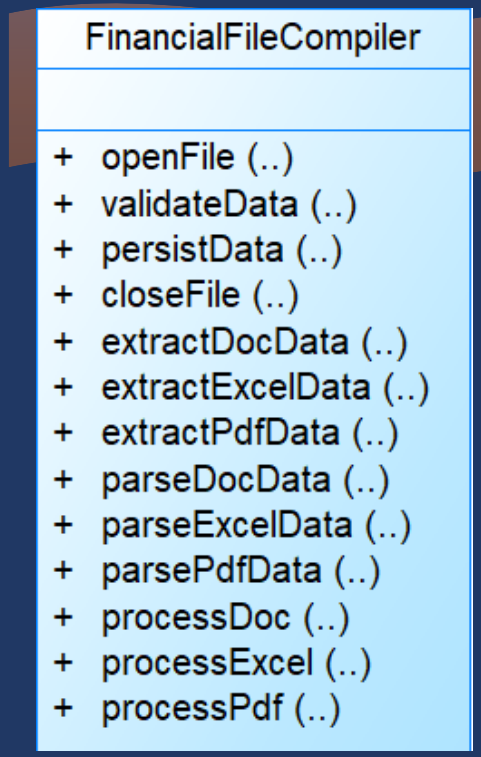
- *Strategy*
- *State*
- *Observer*



Dizajn šabloni

Strategy

State



šta menjamo za `backupFile`*?*

FinancialFileCompiler
{abstract}

processFinancialDoc (..)
openFile (..)
extractFinancialData (..)
parseFinancialData (..)
validateData (..)
persistData (..)
closeFile (..)

zar nije nepromenljivo?

Nepromenljivo

FinancialDocCompiler
<<Override>> extractFinancialData (..)
<<Override>> parseFinancialData (..)

FinancialExcelCompiler
<<Override>> extractFinancialData (..)
<<Override>> parseFinancialData (..)

FinancialPDFCompiler
<<Override>> extractFinancialData (..)
<<Override>> parseFinancialData (..)

FinancialCsvCompiler
<<Override>> extractFinancialData (..)
<<Override>> parseFinancialData (..)

Promenljivo

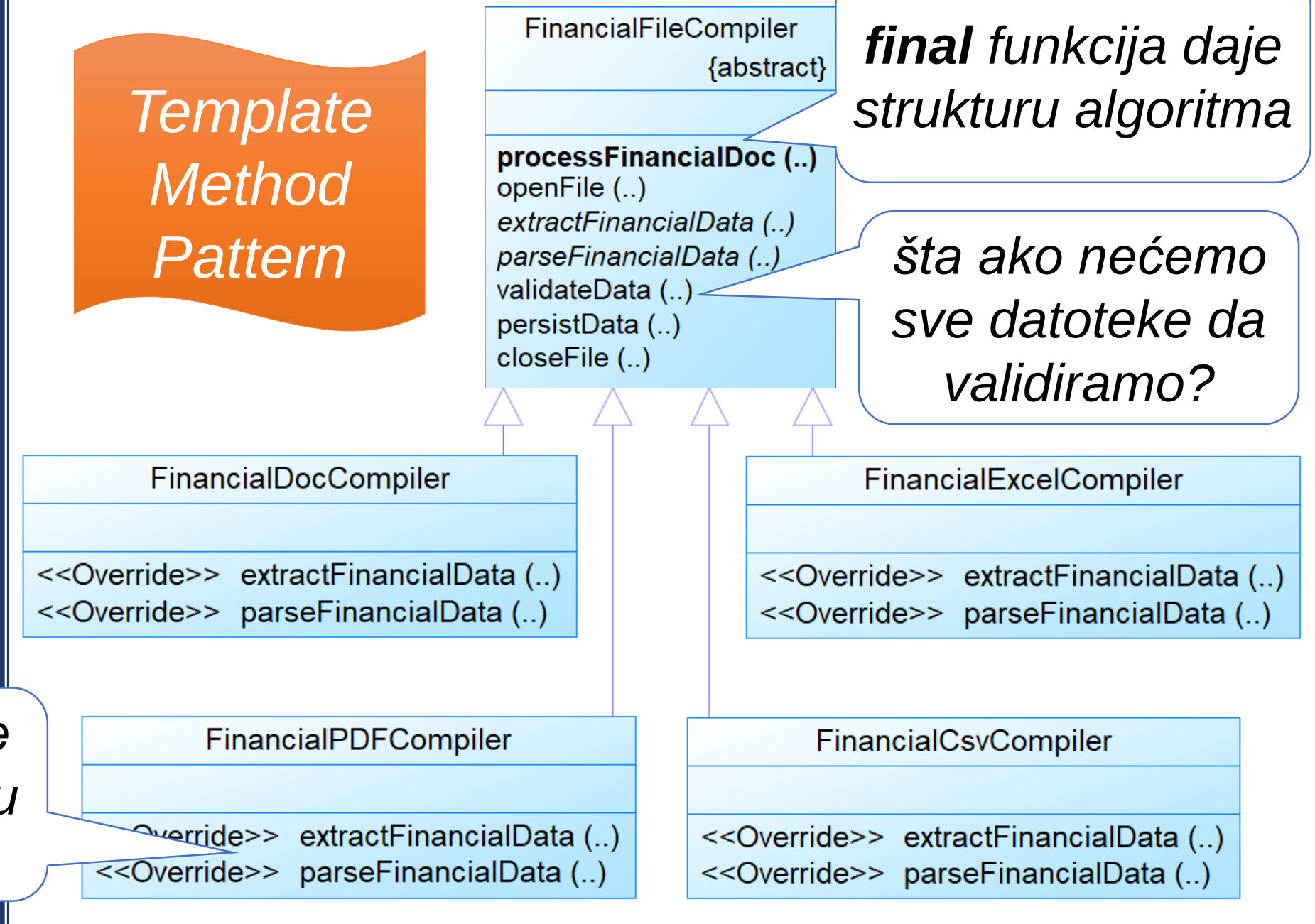
Dizajn šabloni

Strategy

State

Observer

Template Method Pattern



određene korake algoritma definišu naslednice

Dizajn šabloni

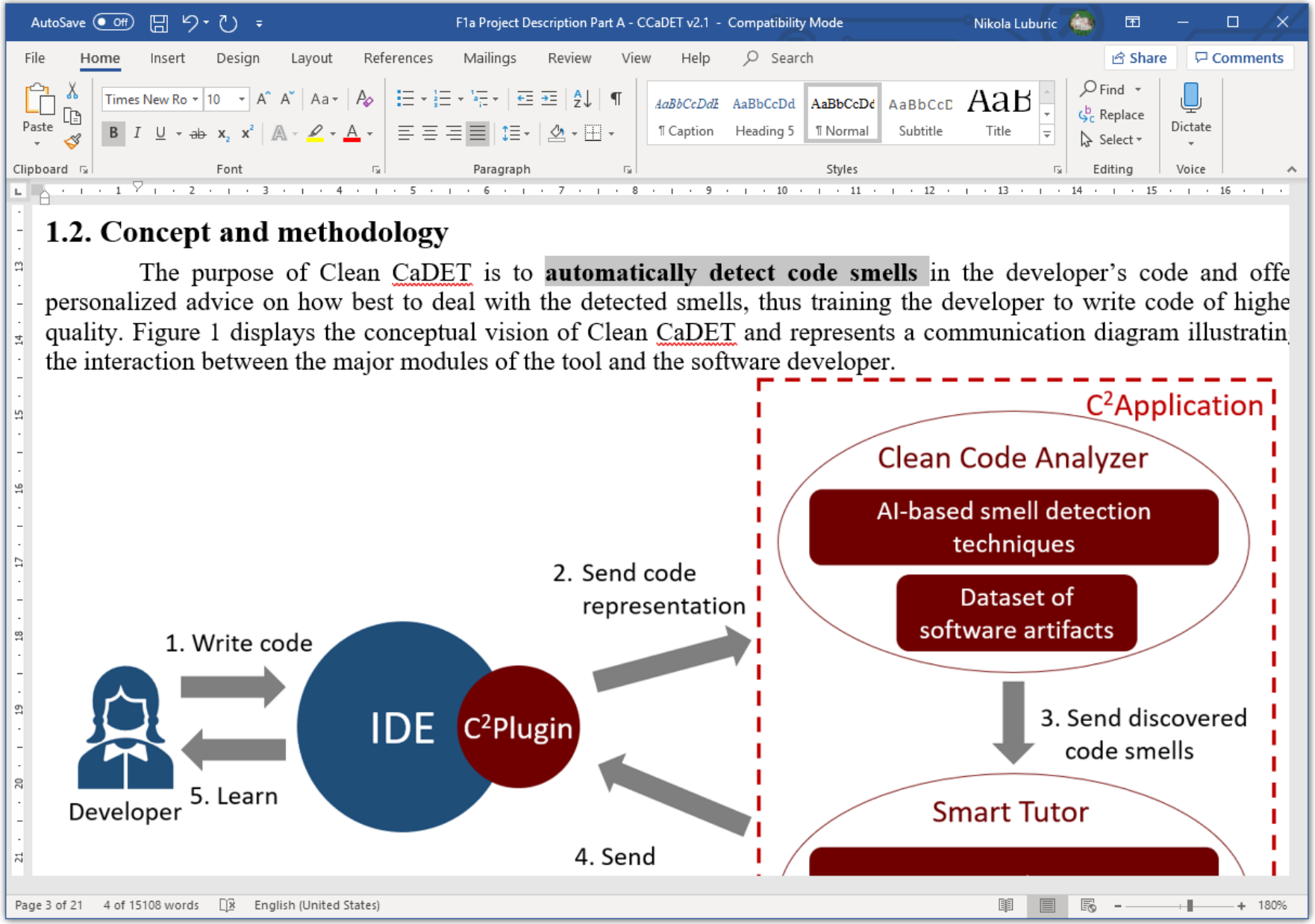
- *Strategy*
- *State*
- *Observer*
- *Template M.*

Definisanje strukture algoritma, gde se definisanje određenih koraka ostavlja klasama naslednicama

Factory method?

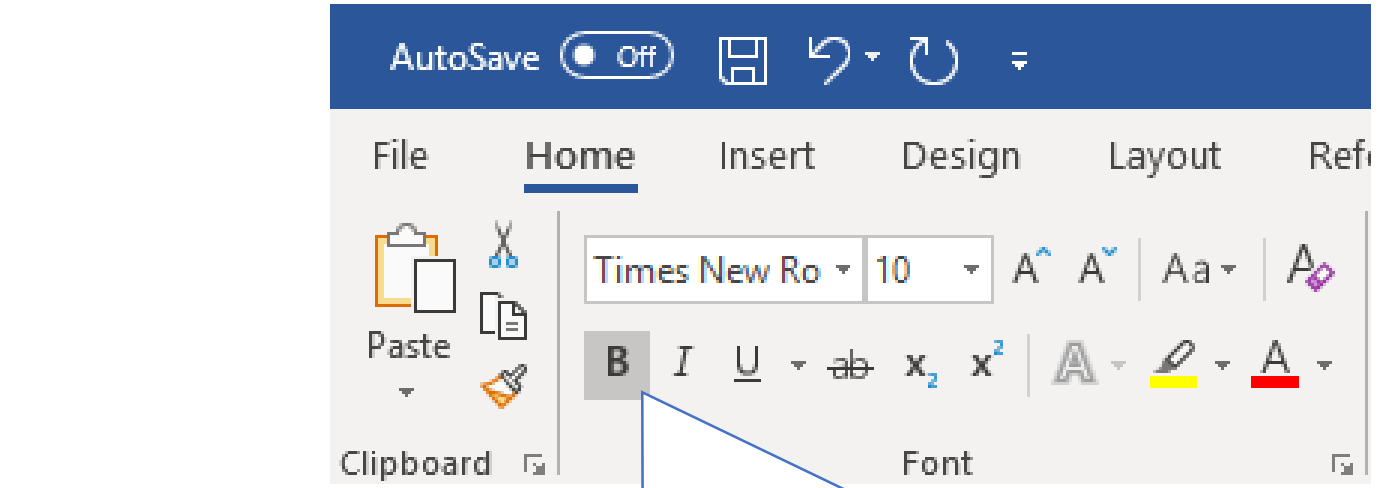
Dizajn šabloni

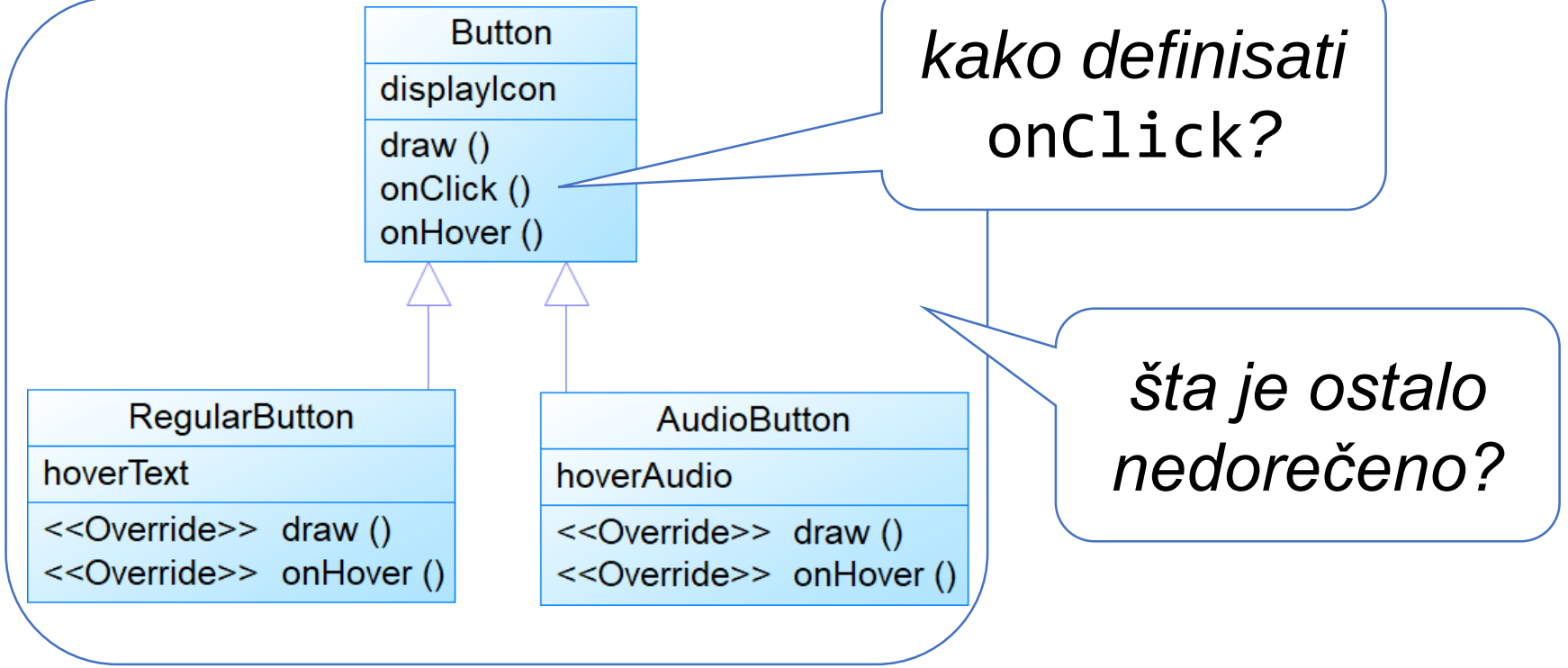
- Strategy
- State
- Observer
- Template M.



Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*



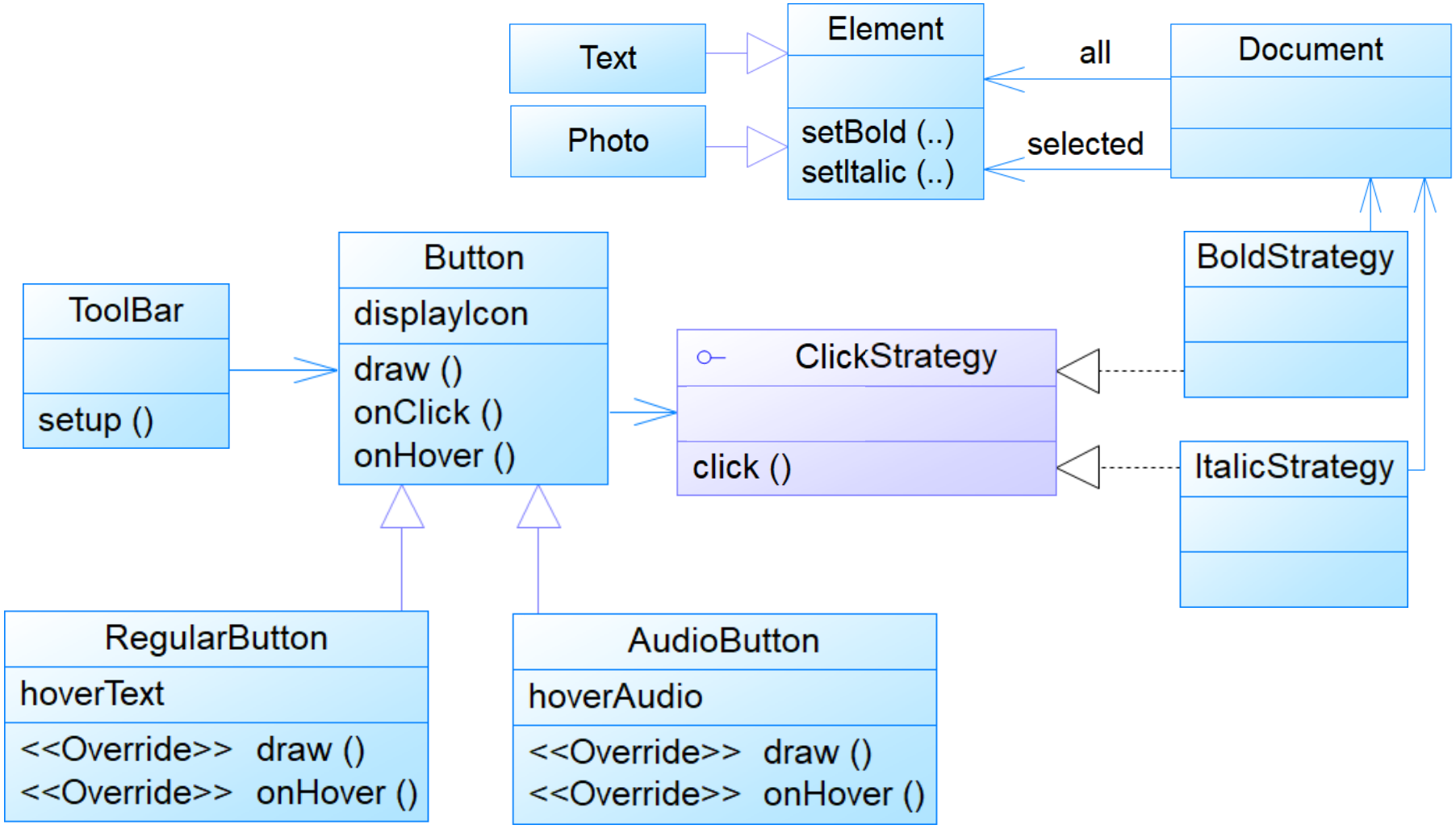


Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*

Napiši pseudokod funkcija i dopuni dijagram:

- `ToolBar > setup() –` za **Bold** i *Italic*
- `BoldStrategy > click()`



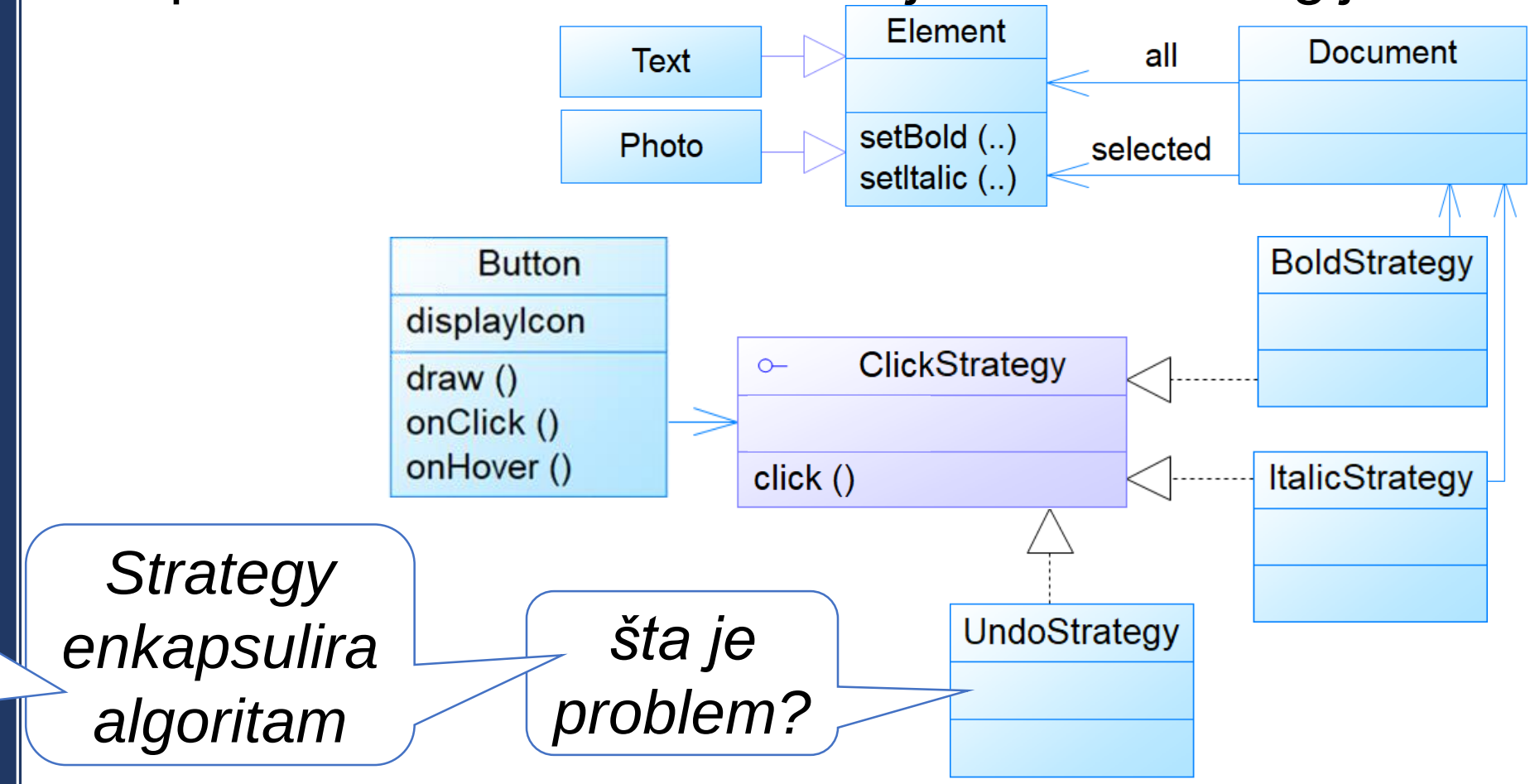
Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*

treba nam enkapsulacija izvršenja algoritma

Definiši proširenje dijagrama i pseudokod funkcije `UndoStrategy > click()` tako da je:

- Moguće uraditi *undo* više uzastopnih operacija
- Ispoštovan OCP za dodavanje novih strategija



Strategy enkapsulira algoritam

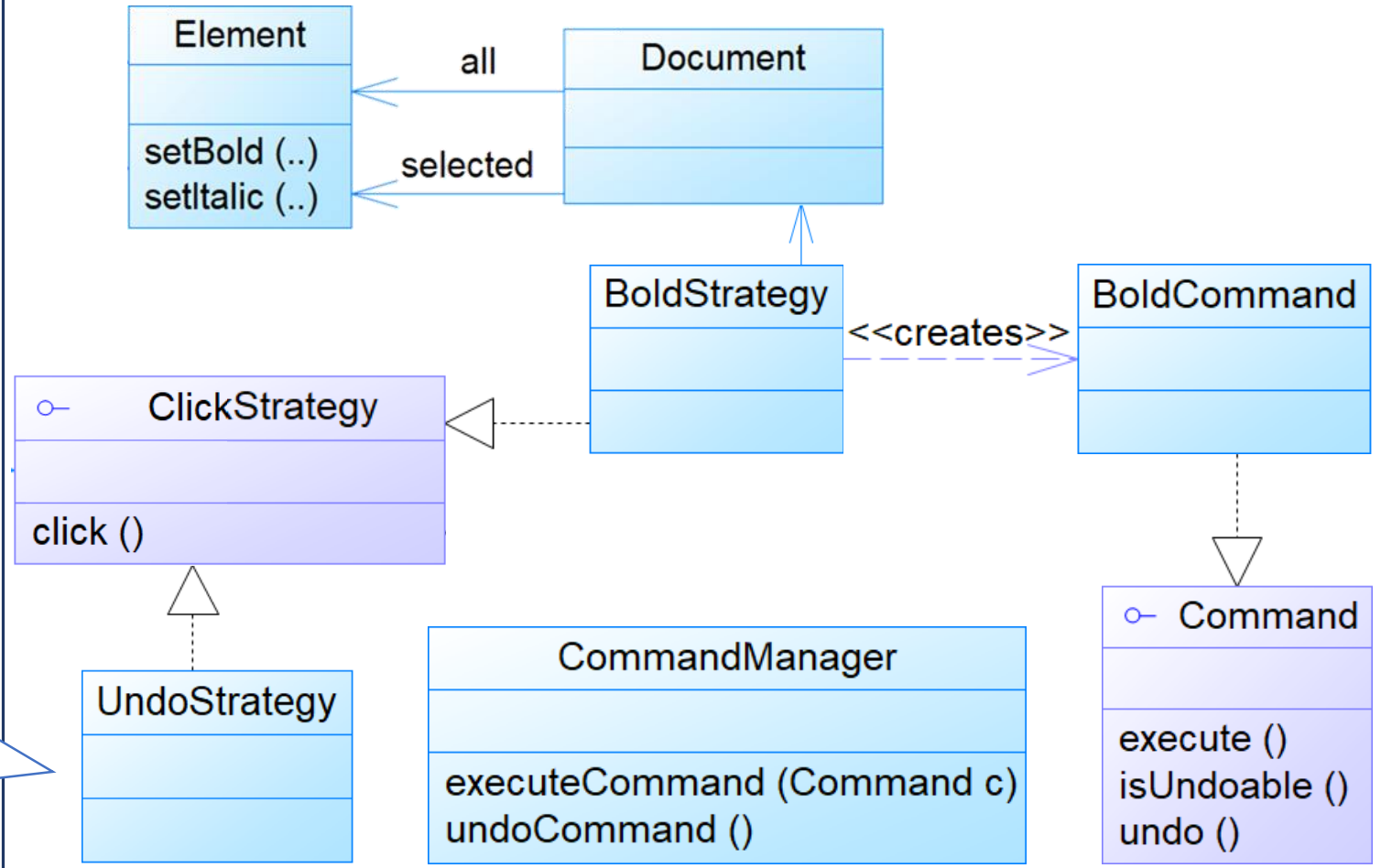
šta je problem?

Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*

treba nam enkapsulacija izvršenja algoritma

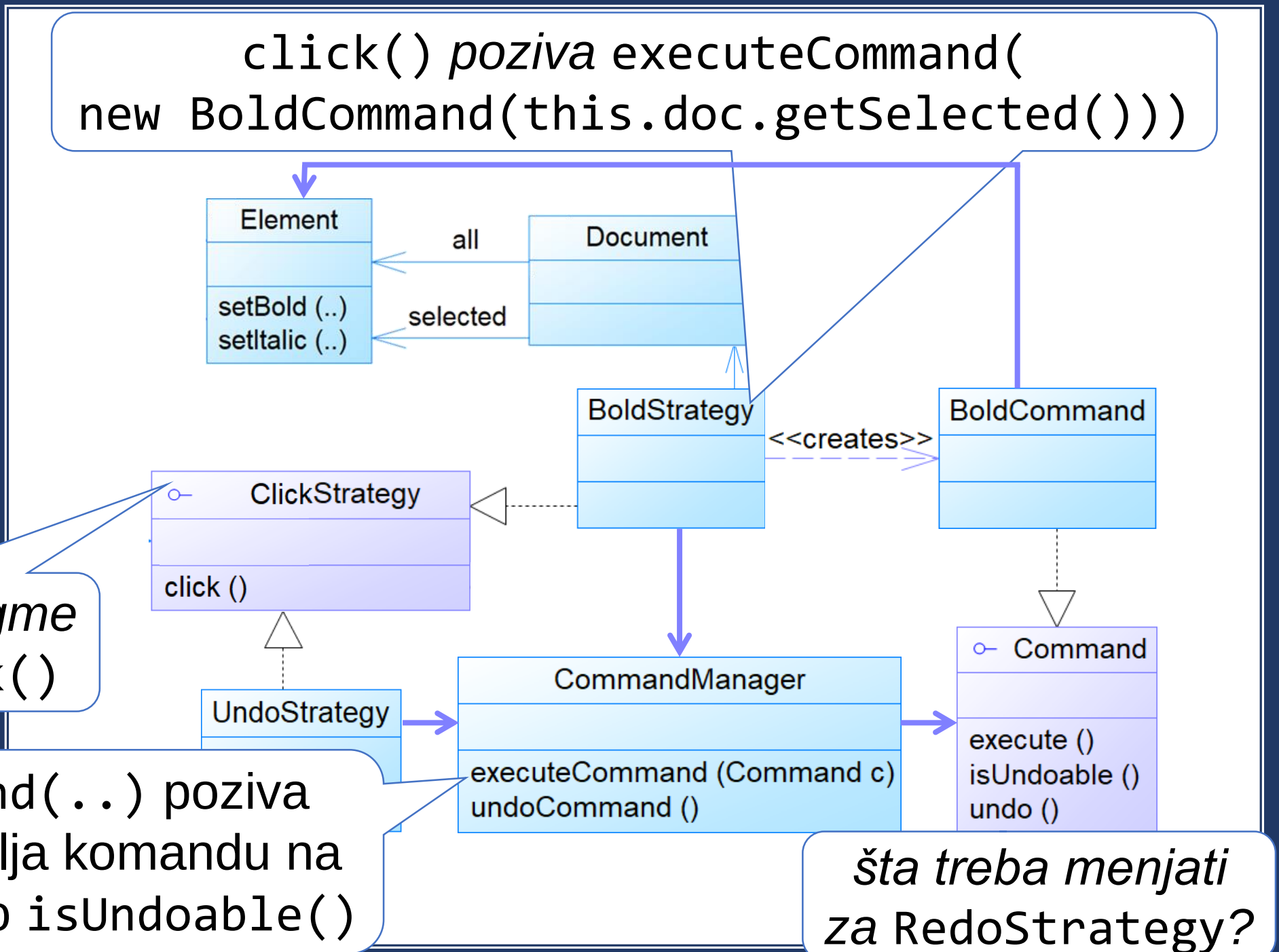
Docrtaj veze između klasa tako da se omogući boldovanja nekog elementa i undo te operacije



Dizajn šabloni

- Strategy
- State
- Observer
- Te

`click()` *poziva* `executeCommand( new BoldCommand(this.doc.getSelected()))`



click na **B** *dugme poziva* `click()`

`executeCommand(..)` poziva `execute()` i stavlja komandu na undoable stek ako `isUndoable()`

šta treba menjati za `RedoStrategy`*?*

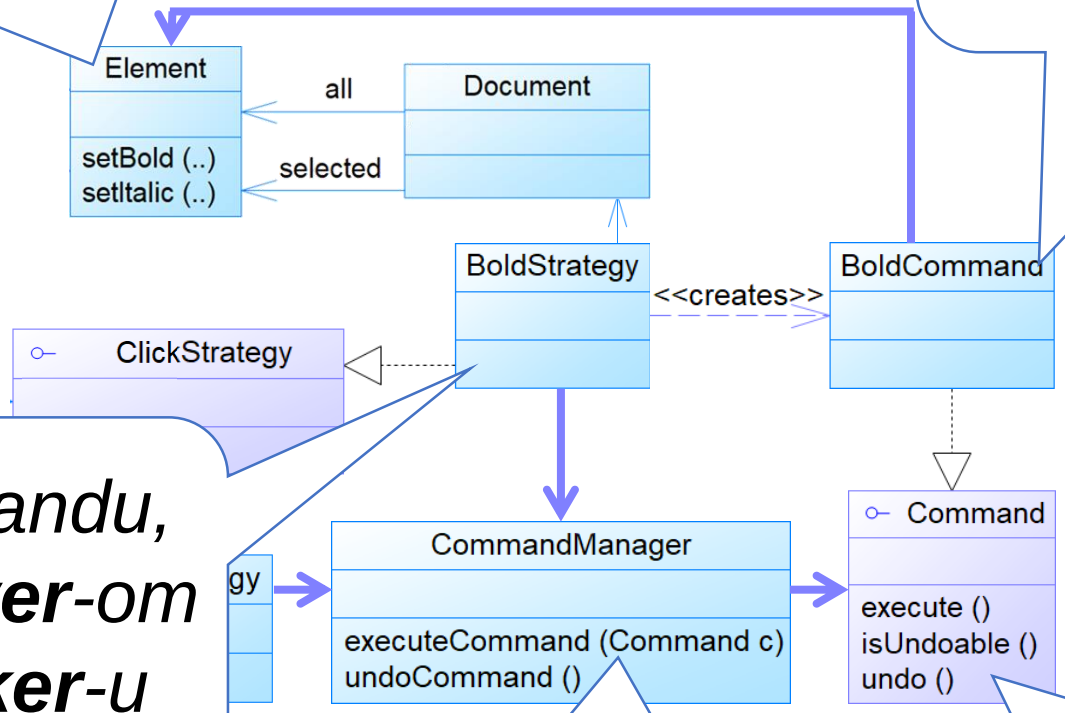
Dizajn šabloni

- Strategy
- State
- Observer
- Te

Command Pattern

Receiver *je objekat koji komanda afektuje*

Konkretna komanda *sadrži informacije i ponašanje vezano za operaciju*



Klijent *kreira komandu, povezuje sa* ***receiver****-om i prosleđuje* ***invoker****-u*

Invoker *aktivira komandu*

Command *interfejs definiše glavne funkcije*

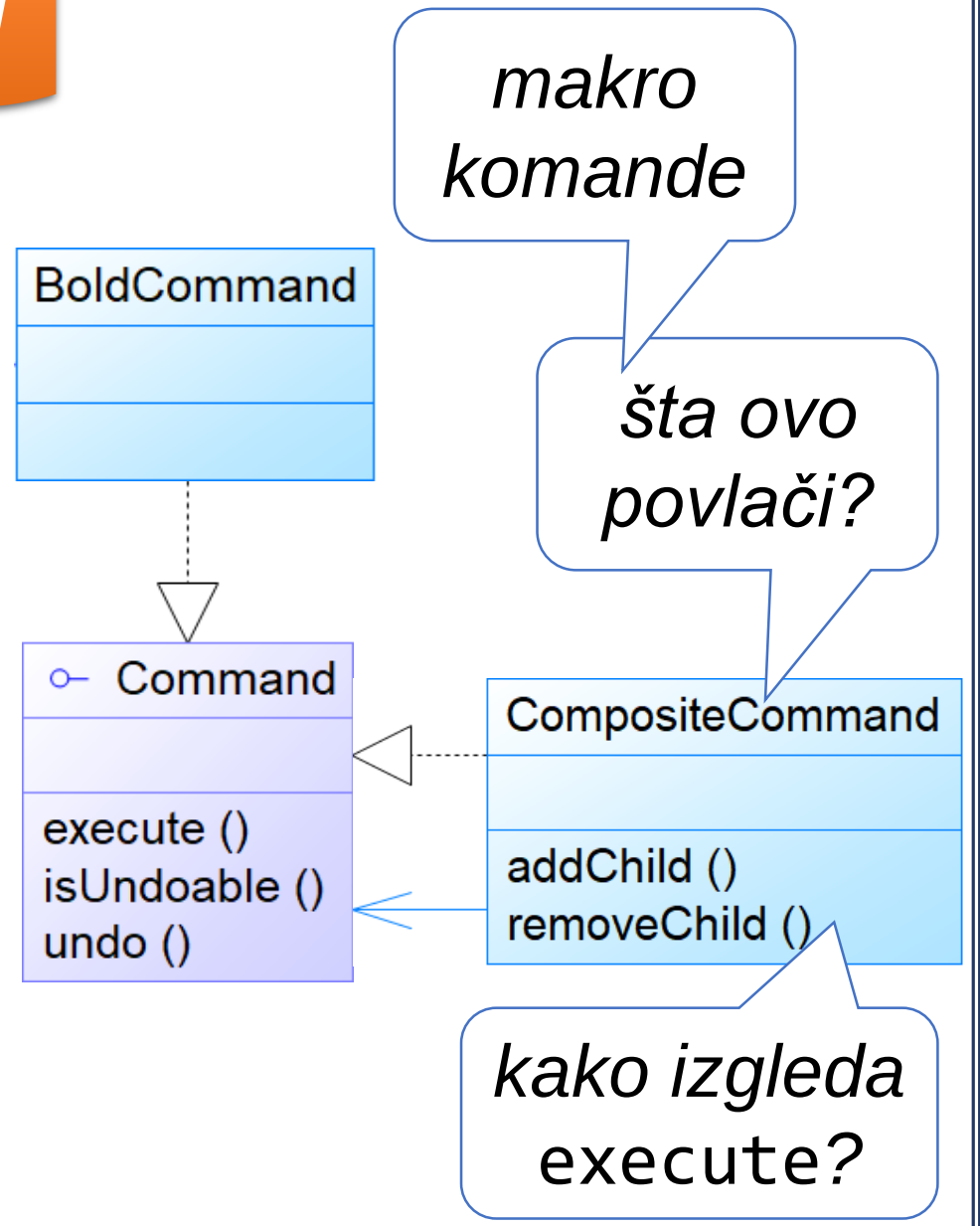
Dizajn šabloni

- Strategy
- State
- Observer
- Template M.

Command Pattern

CompositeCommand
InsertLogo

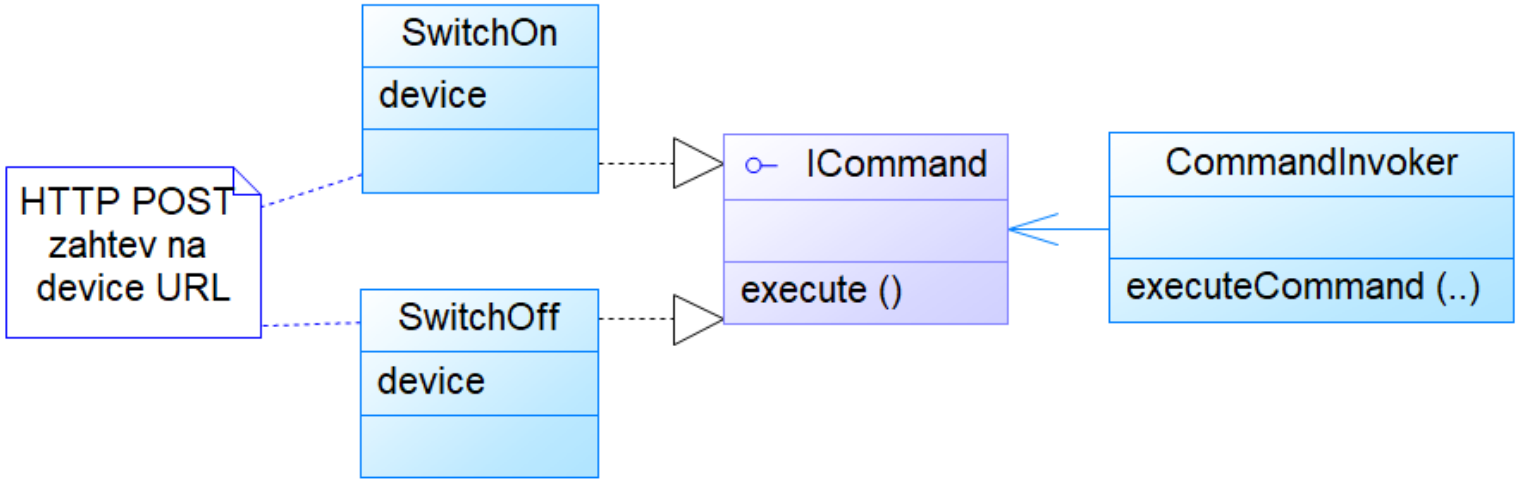
- AddImageCommand
  ftn-logo.png
- AddTextCommand
  FTN
- BoldCommand



Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*
- *Command*

Operateri Elektrovojvodine izdaju komande ka trafostanicama da manipuliše energetskom mrežom.



Može se desiti da komande ne stignu do trafostanice zbog greške u uređaju ili internet mreži.

Z13.4: Napisati pseudokod za odloženo izvršavanje komandi, tako da se sve neizvršene komande izvrše kad proradi mreža (koristite znanje iz Web program.)

Dizajn šabloni

- *Strategy*
- *State*
- *Observer*
- *Template M.*
- *Command*

Enkapsulacija izvršenja operacije u objekat kojim se može manipulisati