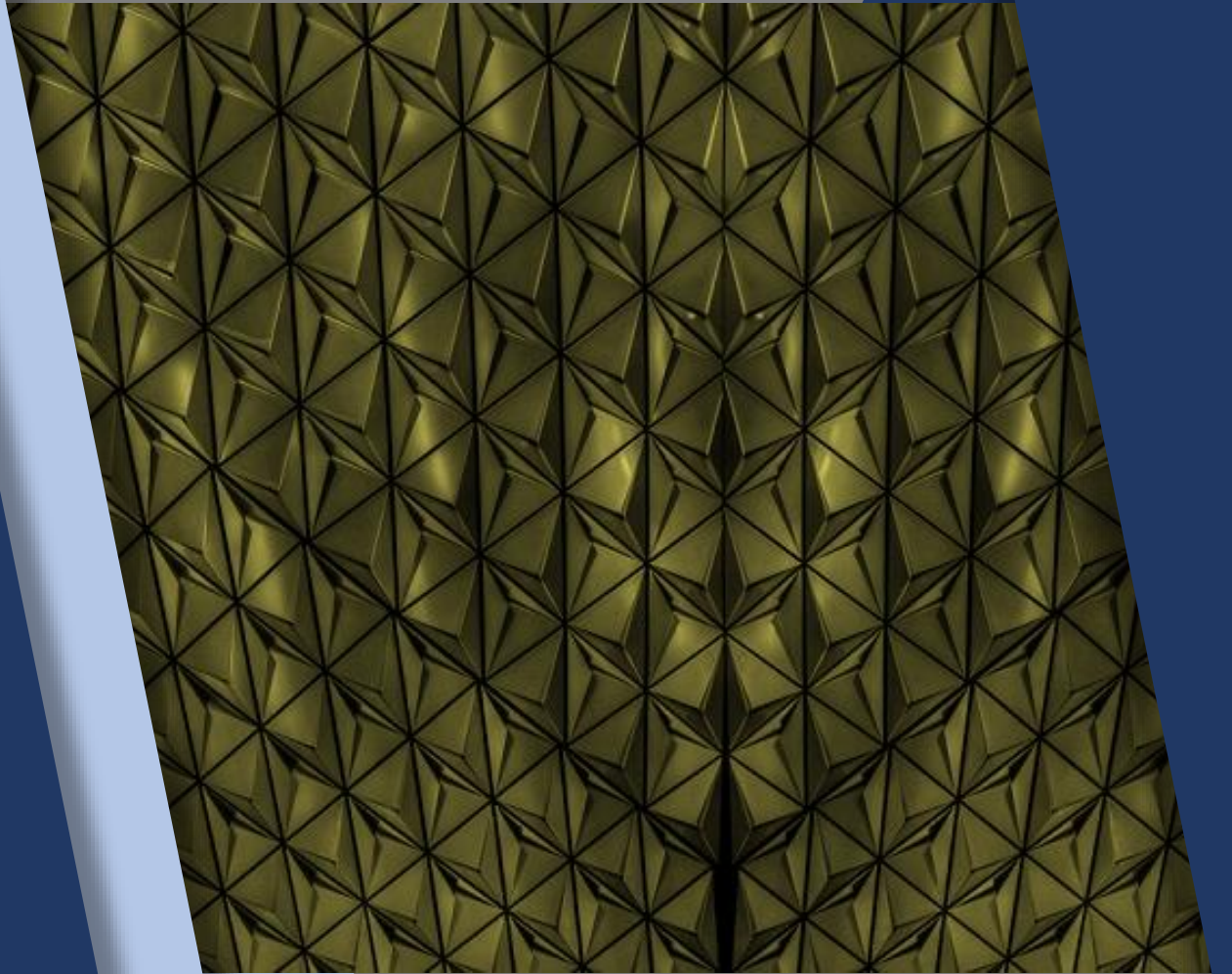


Specifikacija i modeliranje softvera

Dizajn
šabloni

Nikola Luburić
nikola.luburic@uns.ac.rs



Dizajn šabloni

Duck Simulator

Ducks Fed 3

*kako izgleda
objektni model
patke?*

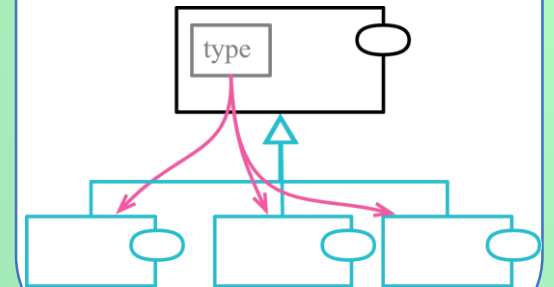


Dizajn šabloni

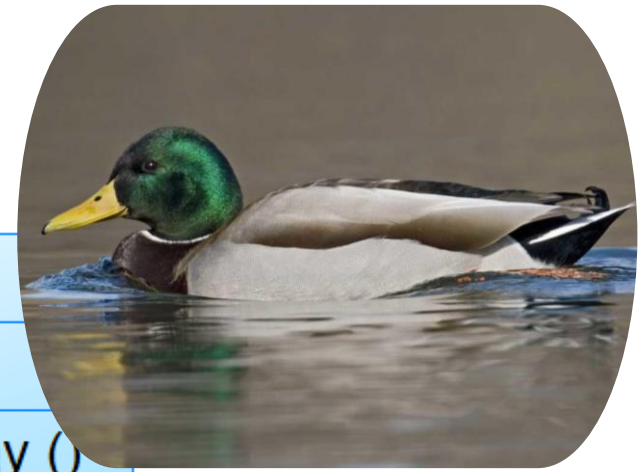
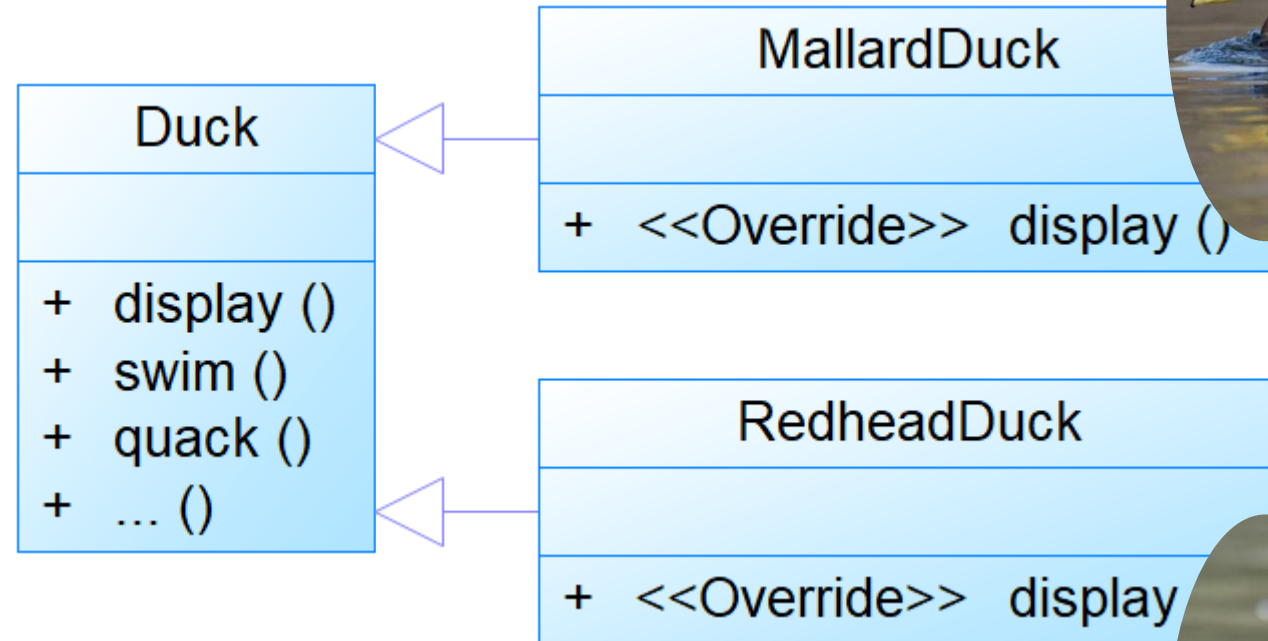
Duck Simulator

Duck	
+	display ()
+	swim ()
+	quack ()
+	... ()

*display f-ja
za više vrsti?*



Dizajn šabloni



*patke treba
da lete*

Dizajn šabloni

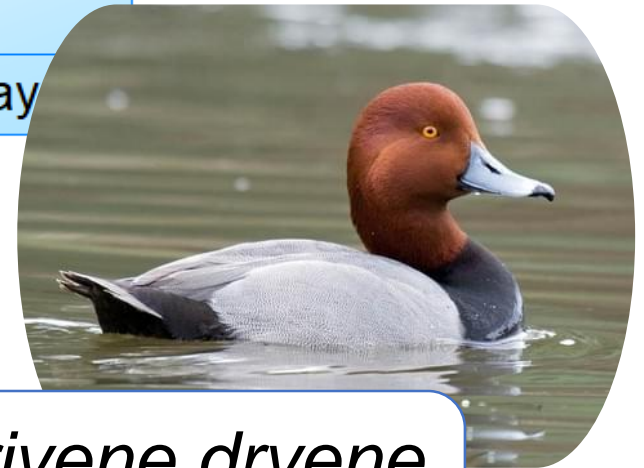
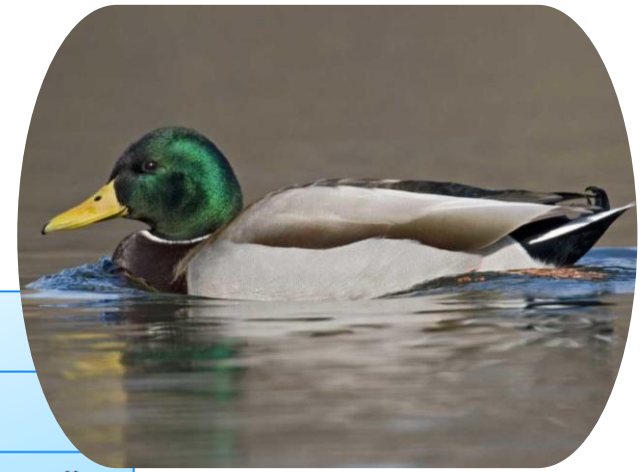
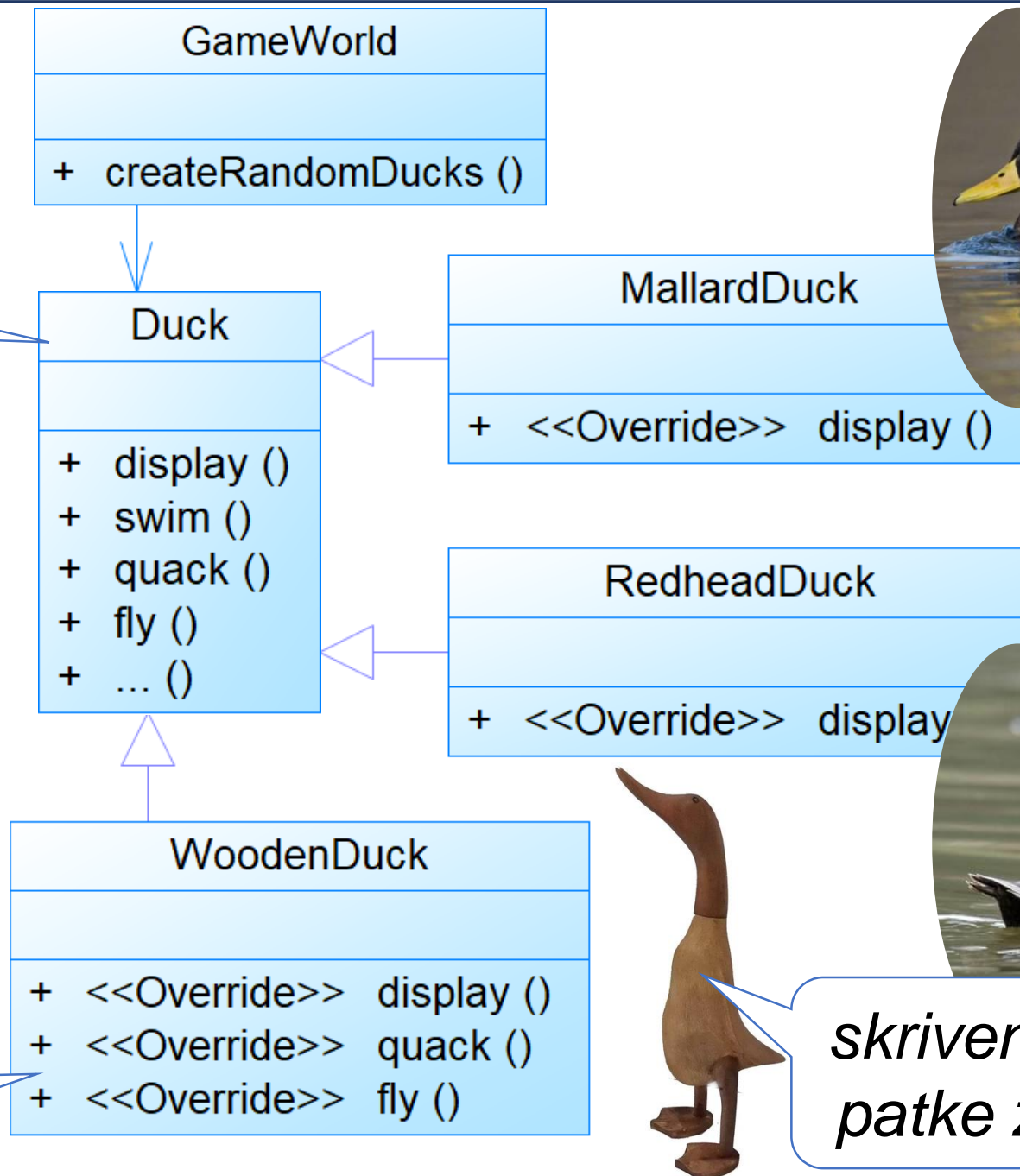
SRP?

*ako nahranjene
Mallard patke
neće da lete?*

*šta ako ipak
hoće kad
jedu kokice?*

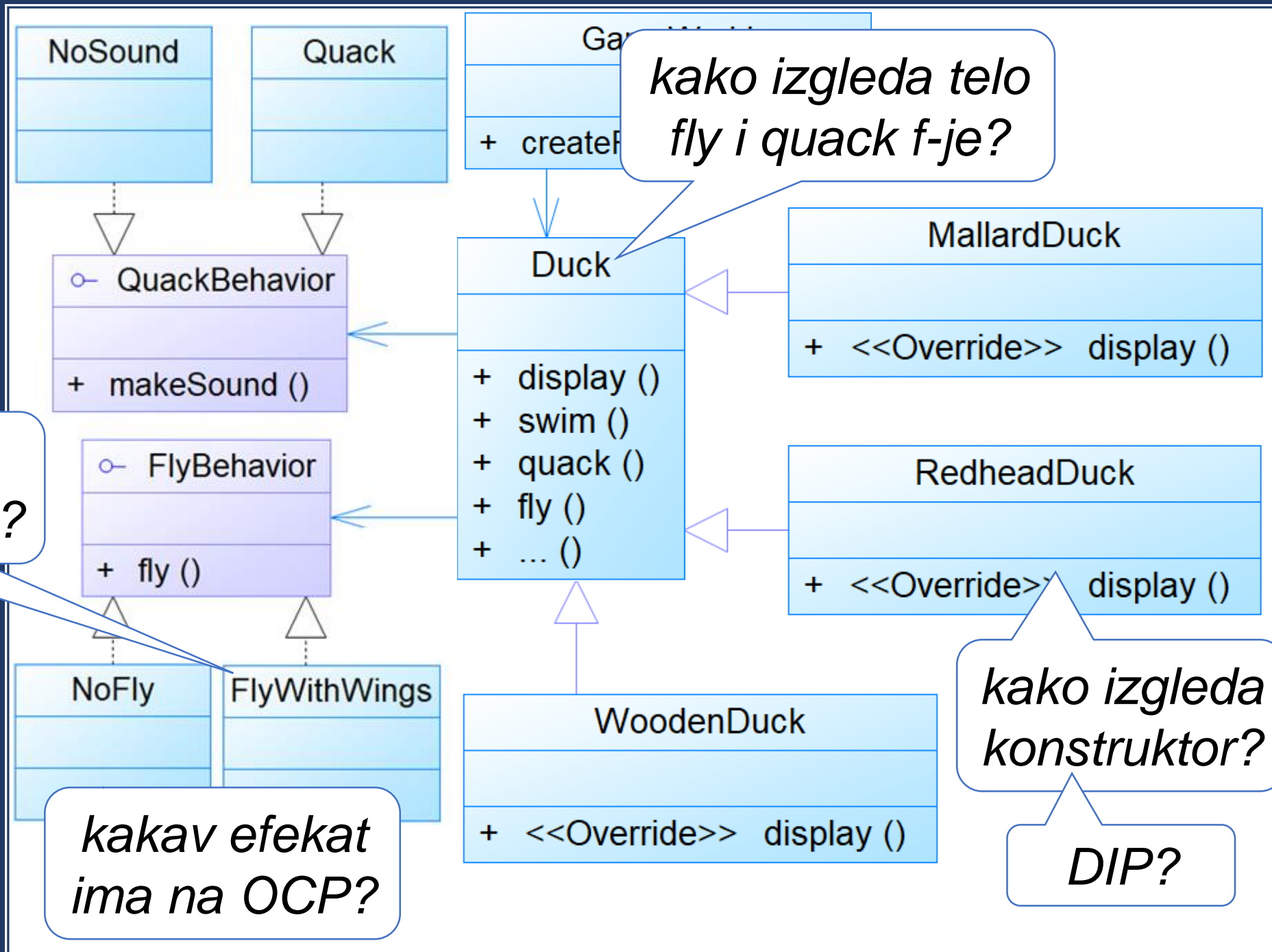
OCP?

LSP?



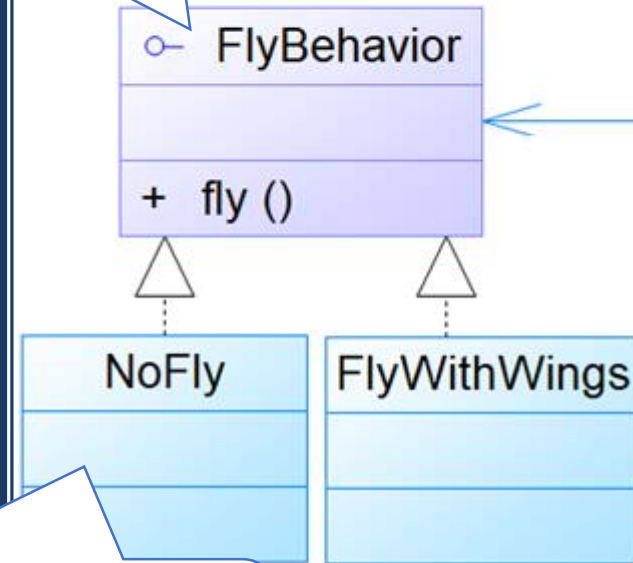
*skrivenne drvene
patke za poene*

Dizajn šabloni



Dizajn šabloni

Strategija je
zajednički interfejs za konkretne strategije



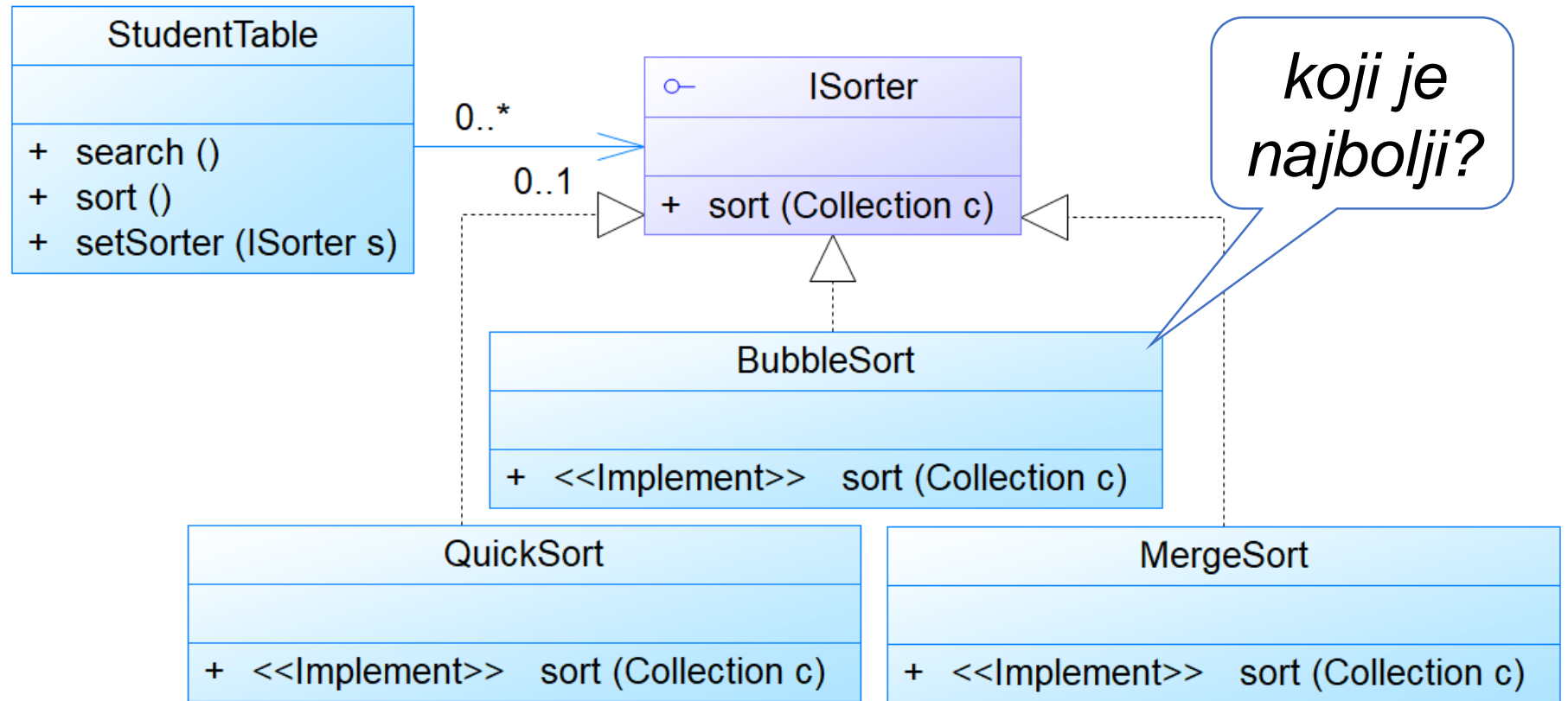
Konkretna strategija je
implementacija algoritma

Klijent
dodeljuje konkretnu strategiju kontekstu

Kontekst
referencira i koristi strategiju kroz interfejs

Strategy Pattern

Dizajn šabloni



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

Strategy Pattern

Dizajn šabloni



Z13.1: Definirati 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

Definisanje familije algoritama i
omogućavanje njihove zamene
bez uticaja na klijenta

*da li je Repository
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!)

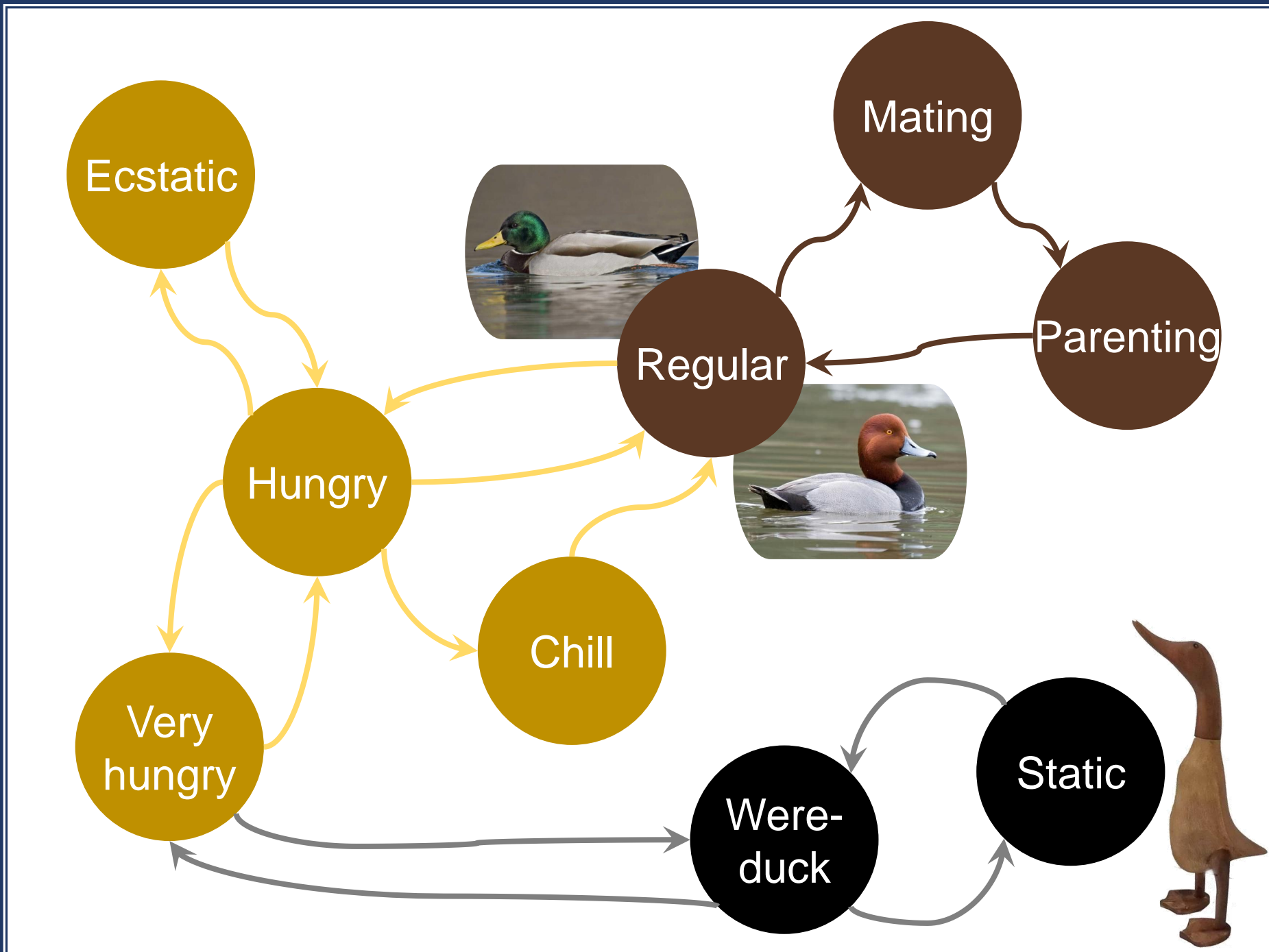
Bonus: Wooden ducks come alive on full moons...

Ducks Fed 3

Dizajn šabloni

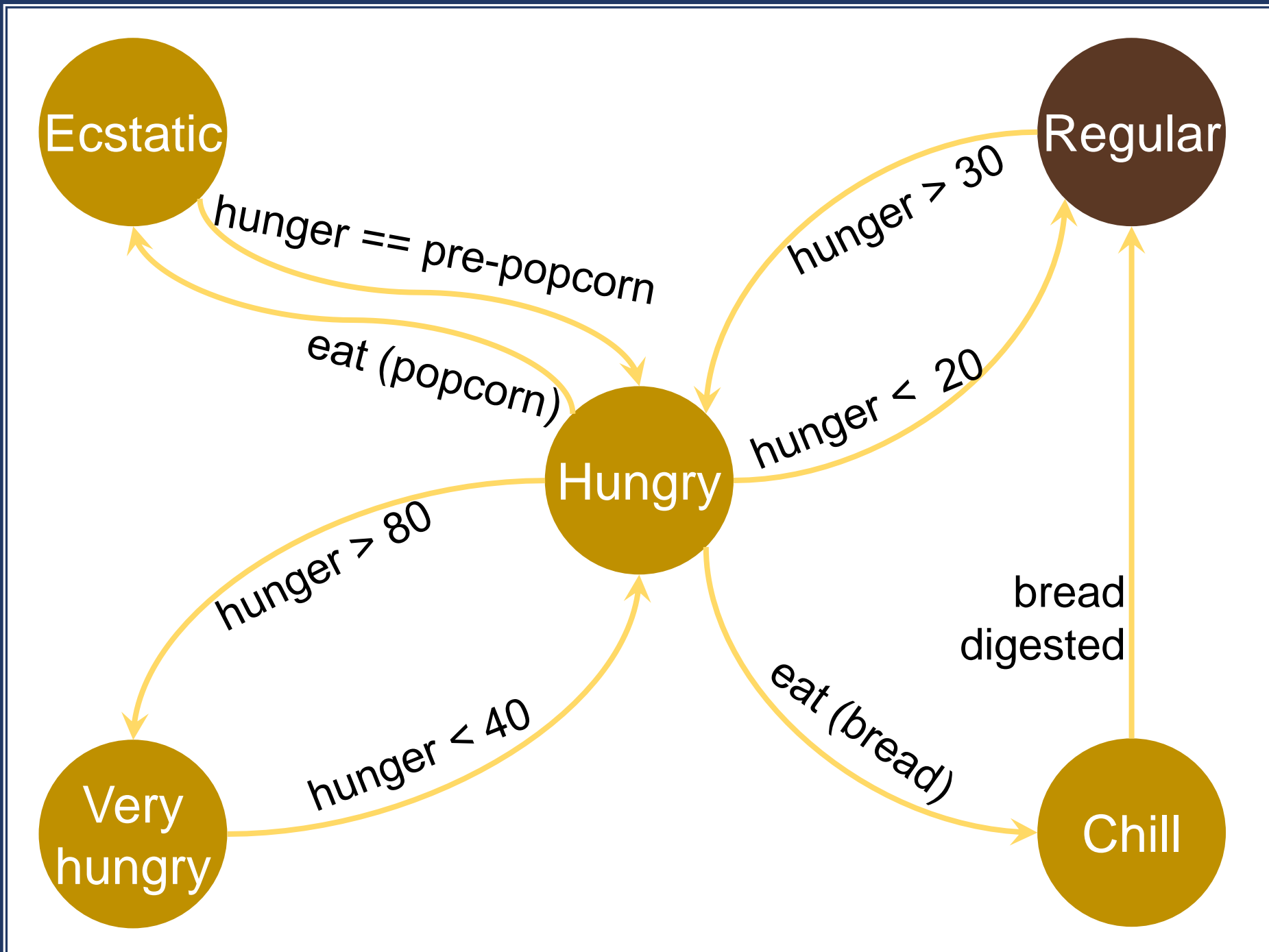
Strategy

šta vidimo?



Dizajn šabloni

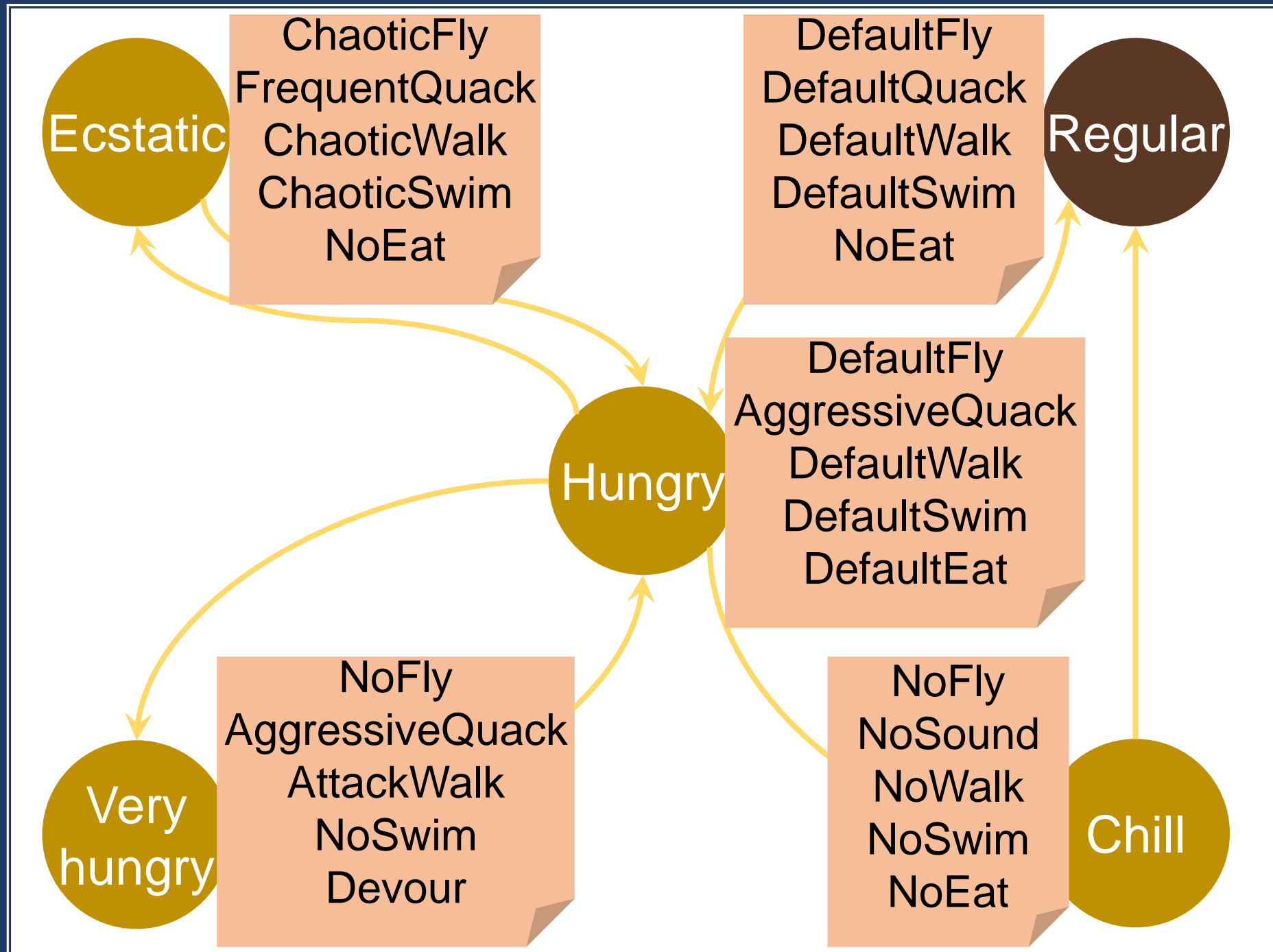
Strategy



Dizajn šabloni

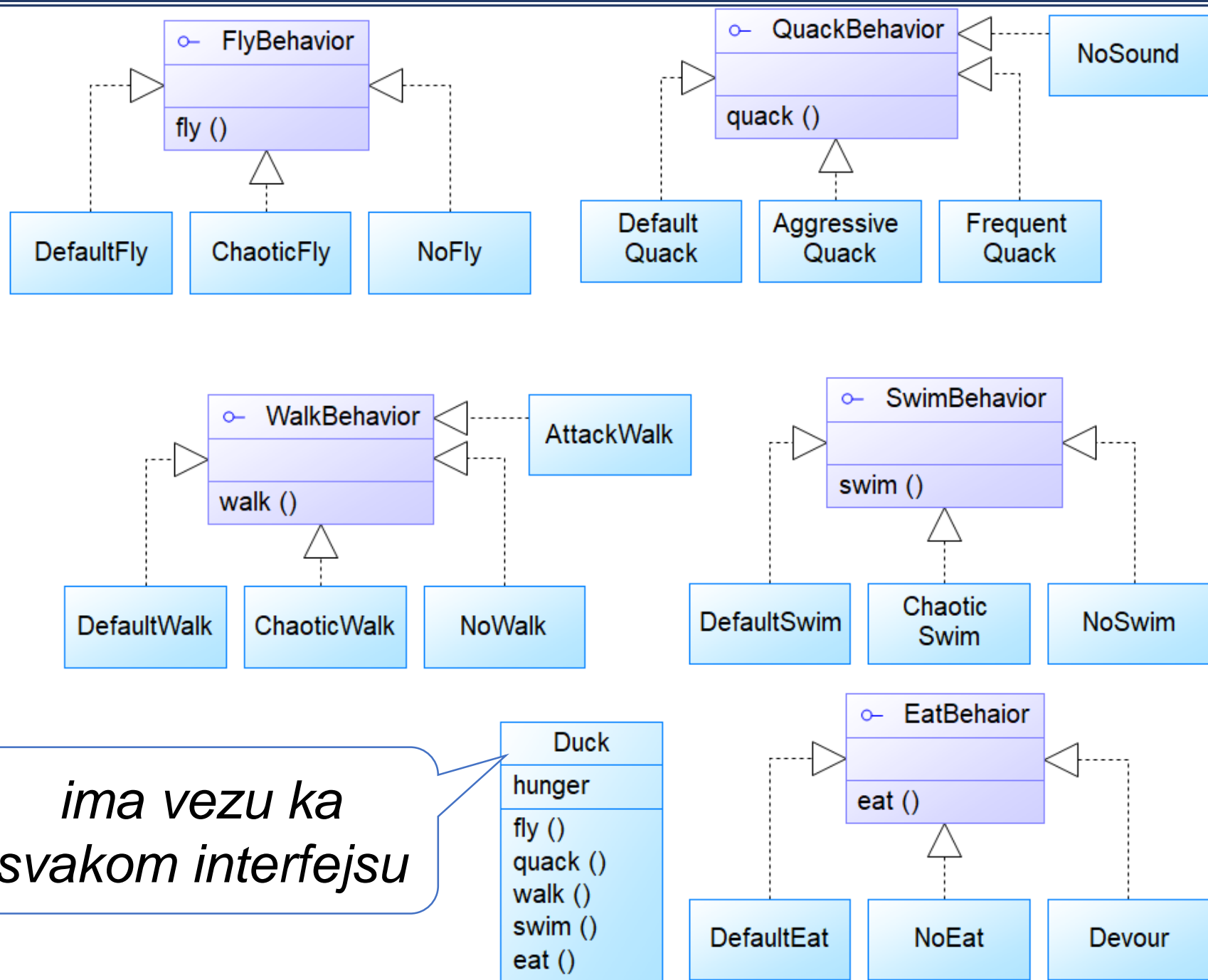
Strategy

*kako izgleda
dijagram klase?*



Dizajn šabloni

Strategy



Dizajn šabloni

Strategy

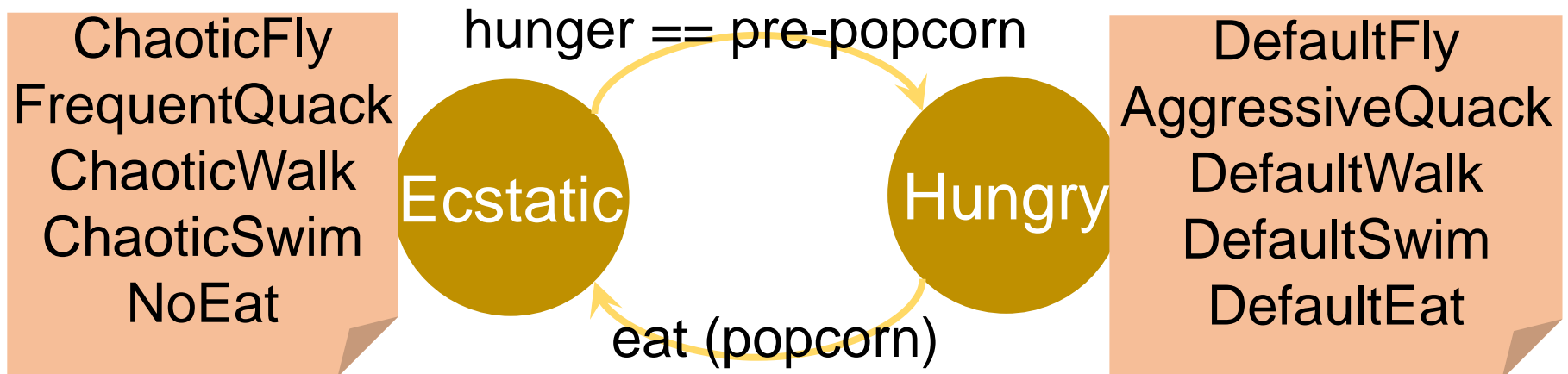
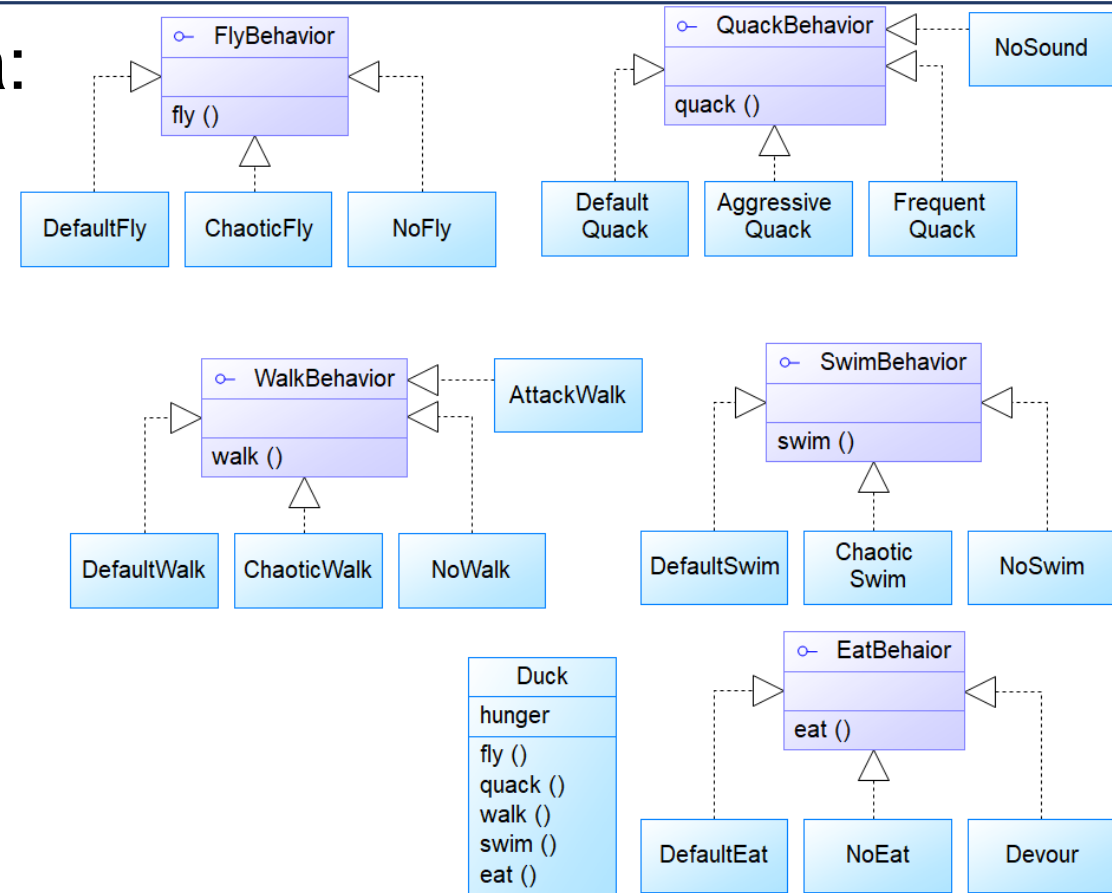
šta je problem?

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*



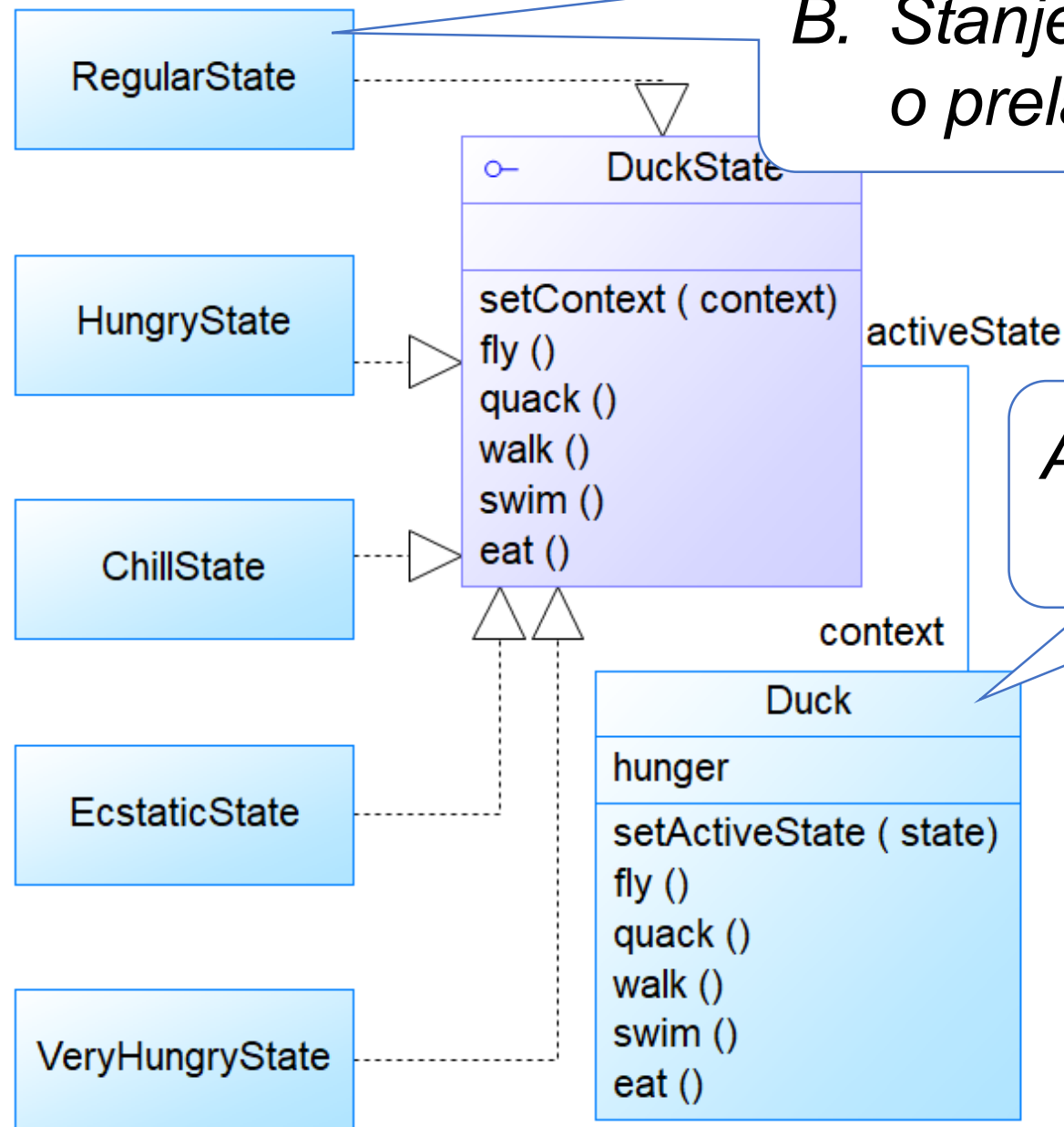
Dizajn šabloni

Strategy

*prednosti
i mane?*

*kako prelazimo iz
stanja u stanje?*

*kako izgleda
kod?*



*B. Stanje brine
o prelasku*

*A. Duck brine
o prelasku*

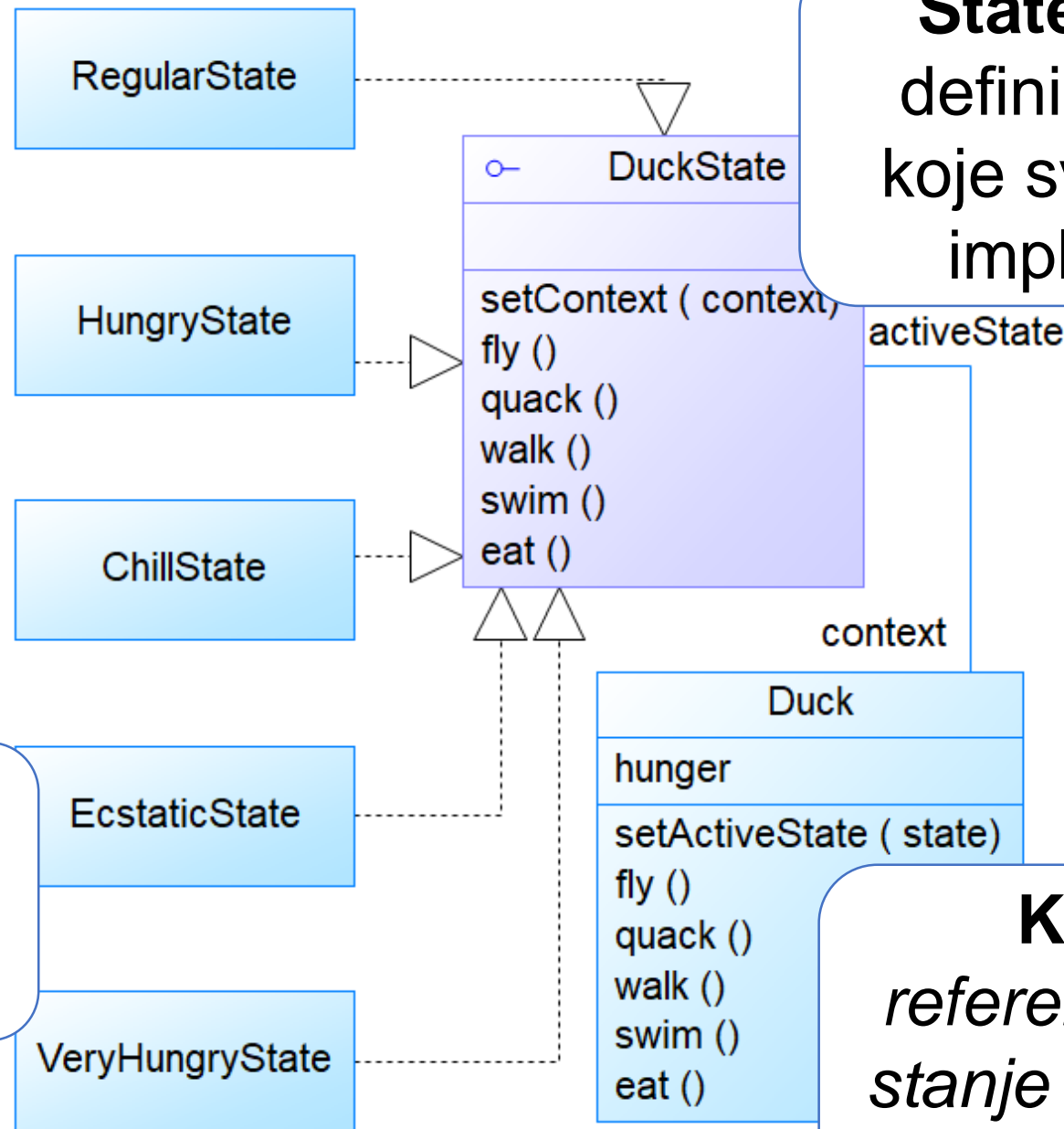
*State
Pattern*

Dizajn šabloni

Strategy

State

Konkretno stanje
implementira state-
specific ponašanje



State interfejs
definiše funkcije
koje svako stanje
implementira

Kontekst
*referencira aktivno
stanje kom delegira
state-specific posao*

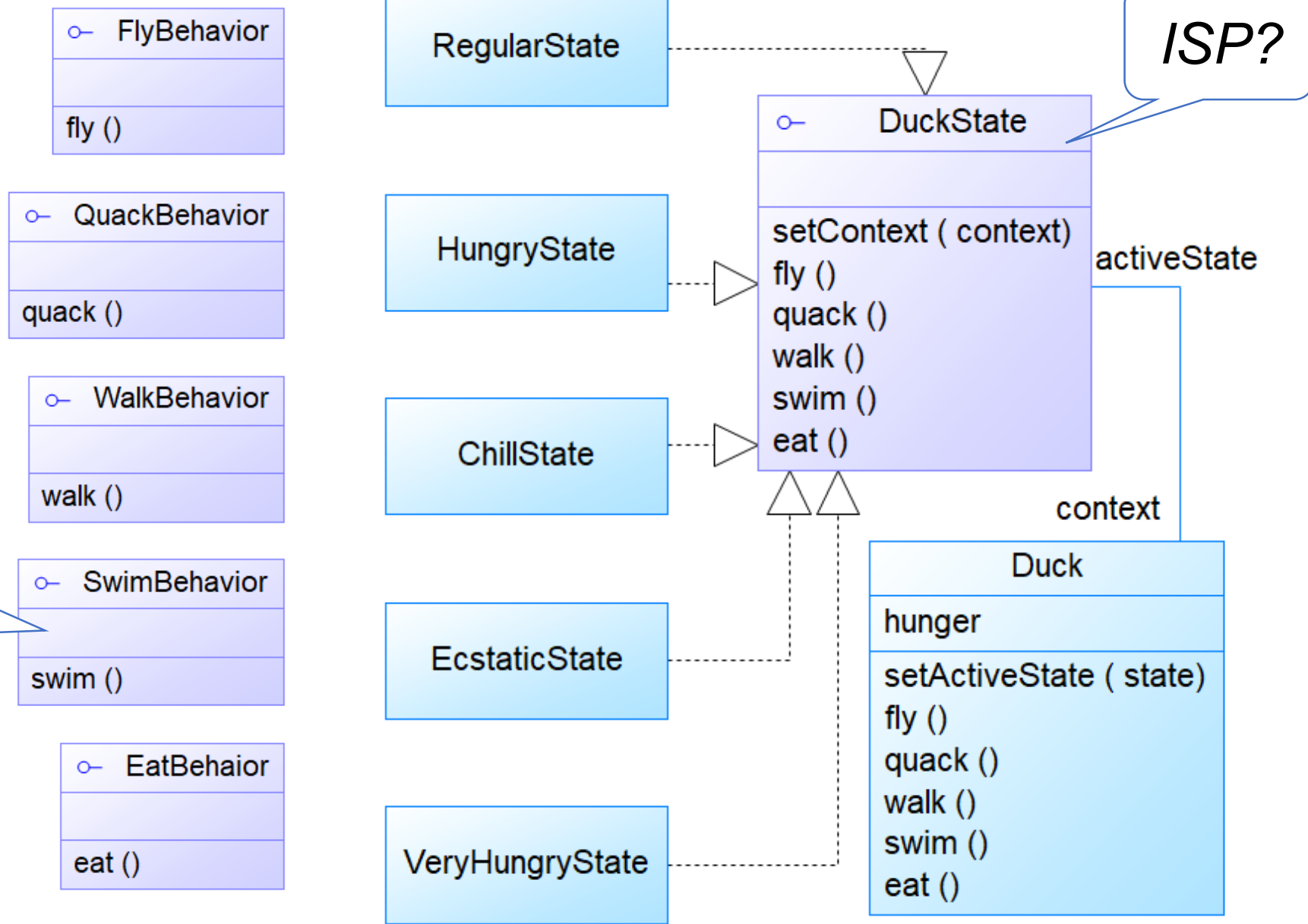
Dizajn šabloni

Strategy

State

možemo
kombinovati
sa strategy

prednosti
i mane?



Strategy

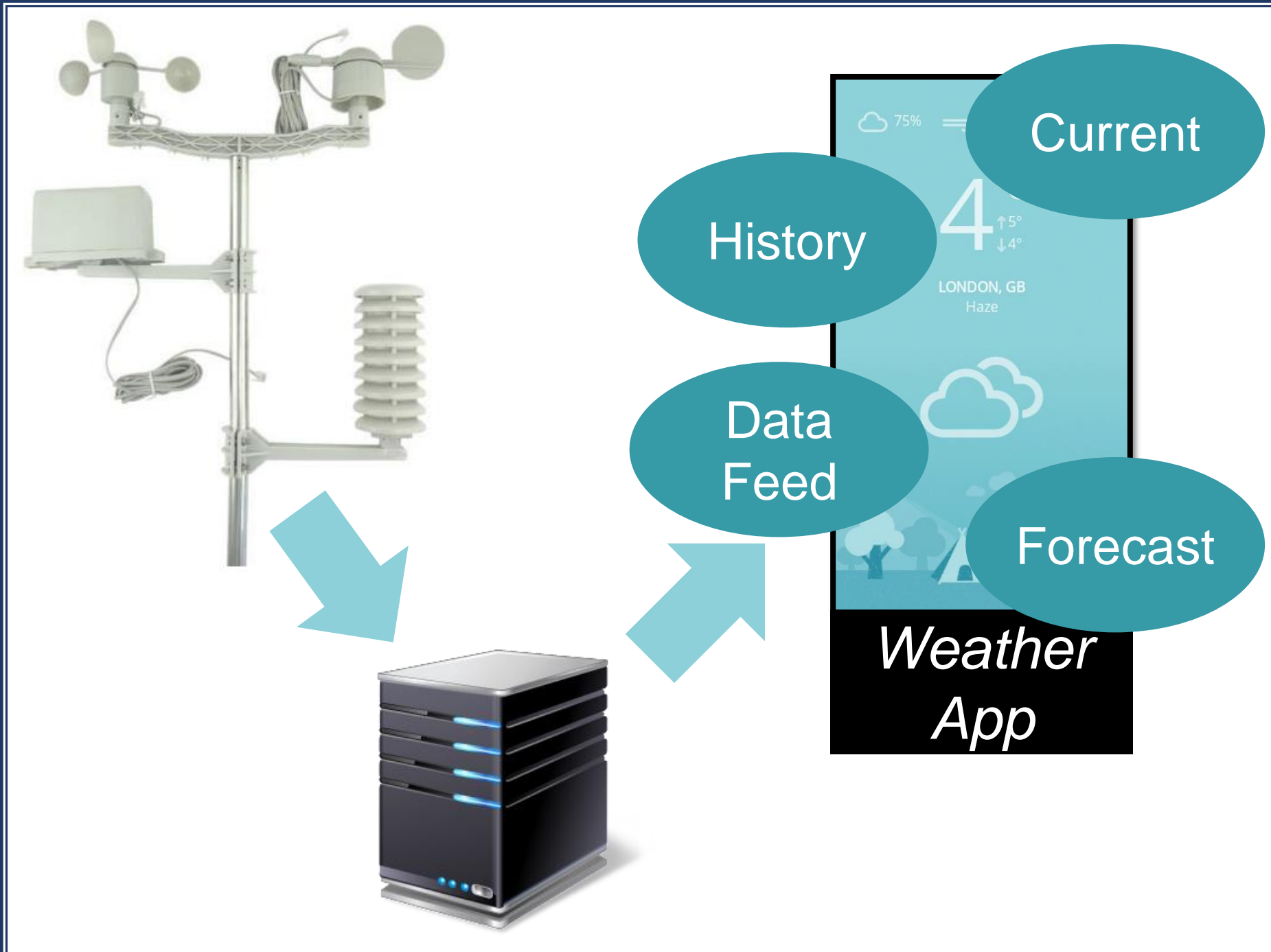
State

Definisanje familije stanja i
omogućavanje izmene
celokupnog ponašanja objekta

Dizajn šabloni

Strategy

State



Dizajn šabloni

Strategy

State

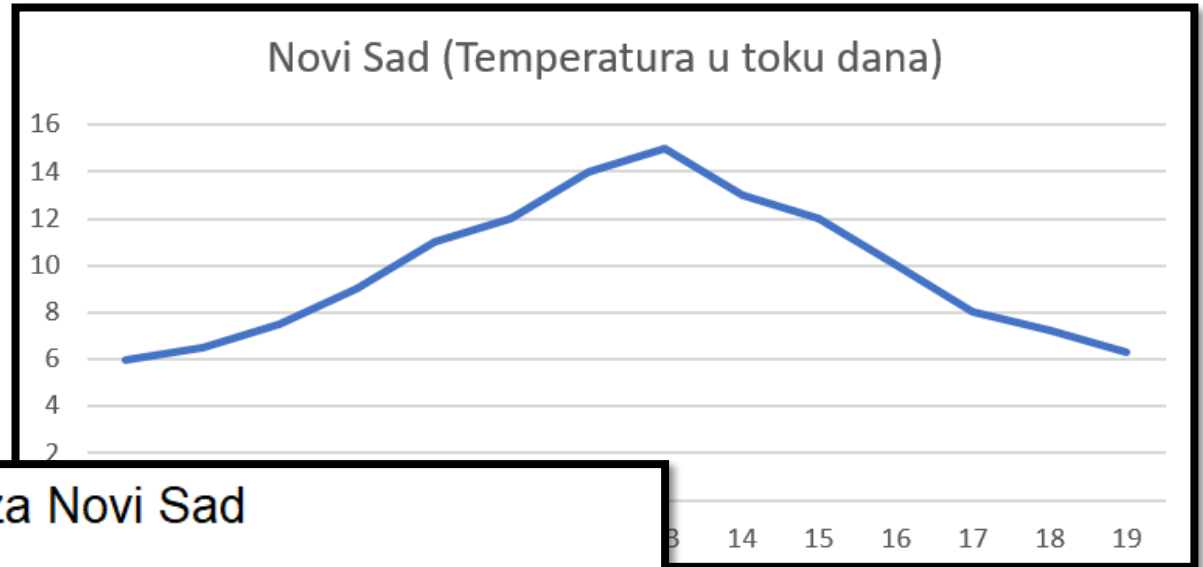
šta kada
stigne novi
podatak?

Data
Feed

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

Current

History



Forecast

Dizajn šabloni

Strategy

State

šta vidimo?

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

Data Feed

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

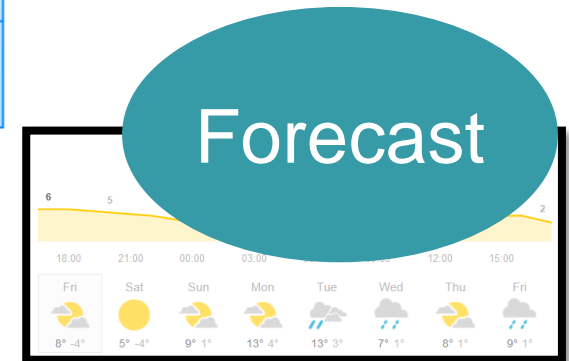
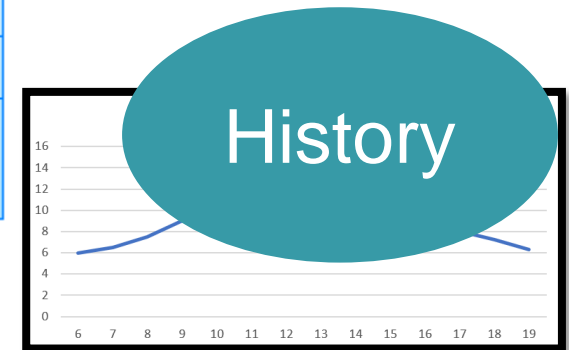
History
- historicalData : double[]
+ updateDisplay ()

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

ForecastStrategy

Current

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



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

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

Novo merenje

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

History
- historicalData : double[]
+ updateDisplay ()

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

ForecastStrategy

zašto ne da svaki modul ima ref. na DataFeed?

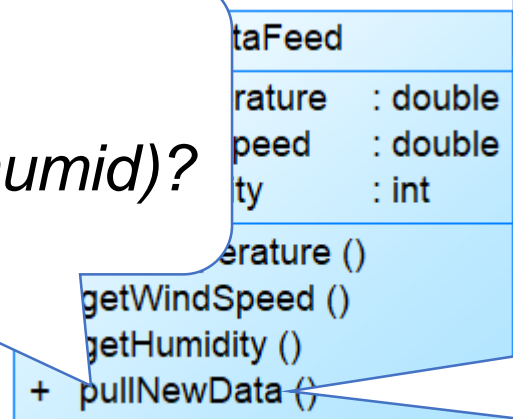
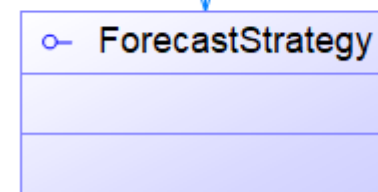
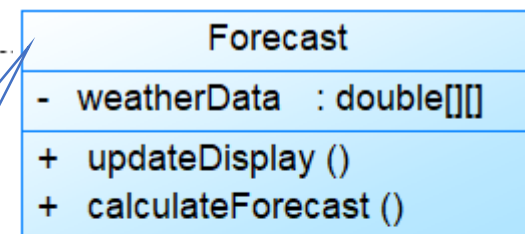
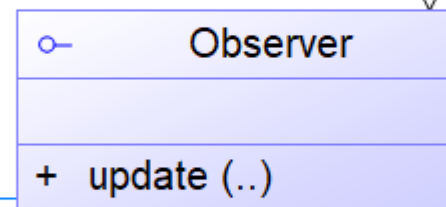
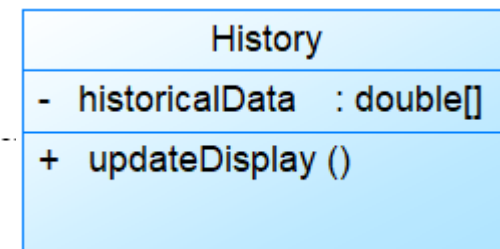
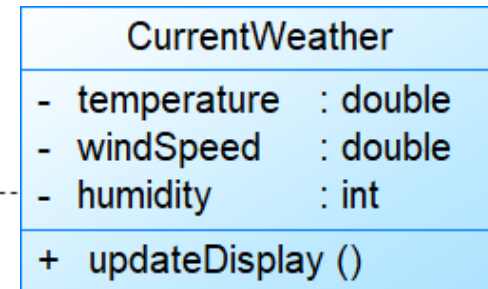
Dizajn šabloni

Strategy

Proširi dijagram i napiši kod da:

- ❖ Svaki modul dobije novo merenje i ažurira prikaz
- ❖ Rešenje mora biti proizvoljno modula

kako izgleda kod?



kako izgleda kod?

kako izgleda kod?

gledaš više feed-ova?

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

proširivost vs
enkapsulacija

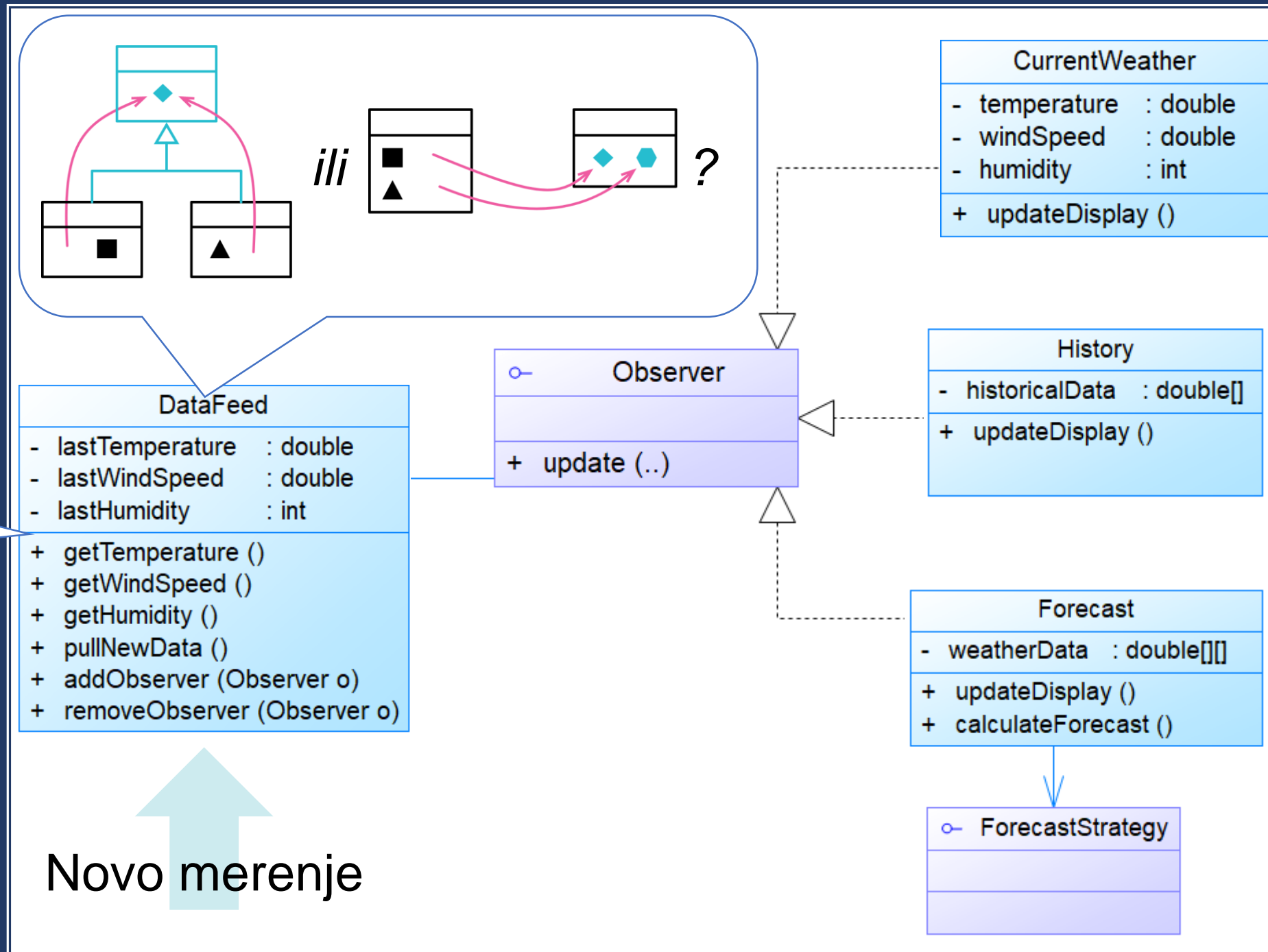
Novo merenje

Dizajn šabloni

Strategy

State

kohezija?



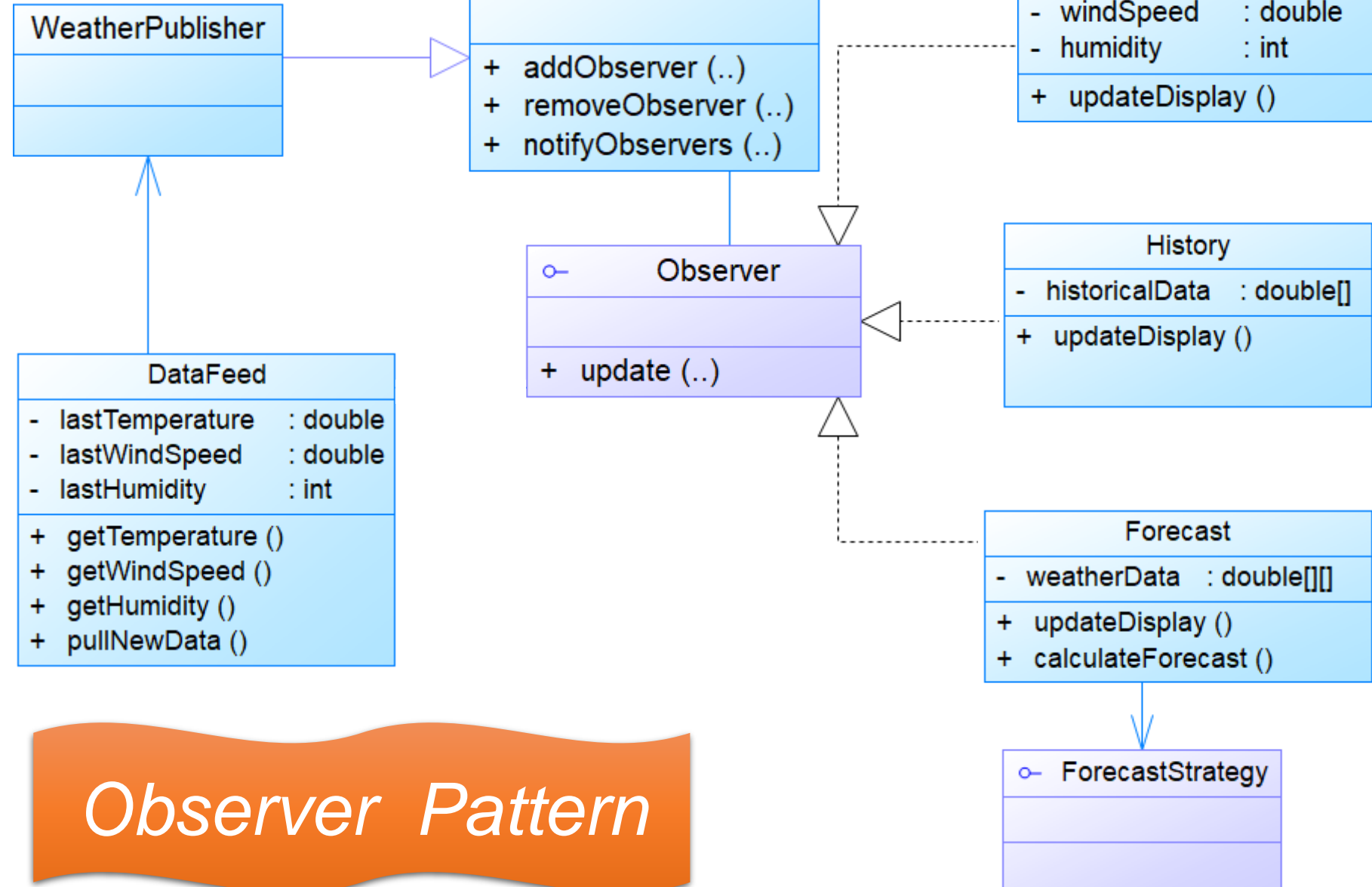
Dizajn šabloni

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



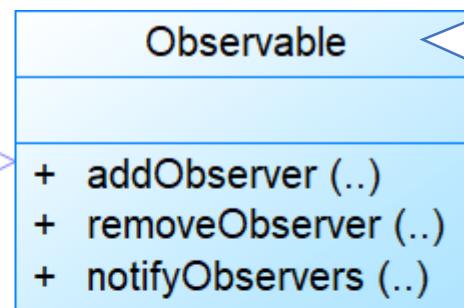
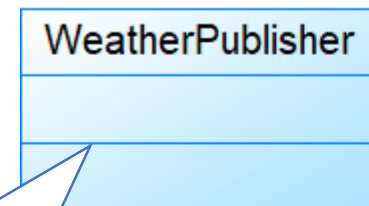
Observer Pattern

Dizajn šabloni

Strategy

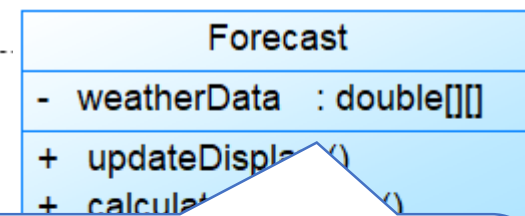
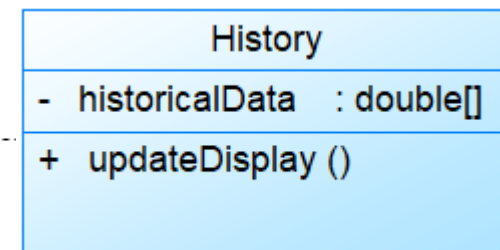
Konkretni subject
obrađuje događaje koje šalje observerima

Z13.3: Istražiti razliku između observer i PubSub mehanizma



Observer koristi
Subject *interfejs za prijavu i odjavu*

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



Konkretni observer se registruje na događaje i reaguje na njih

Observer Pattern

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



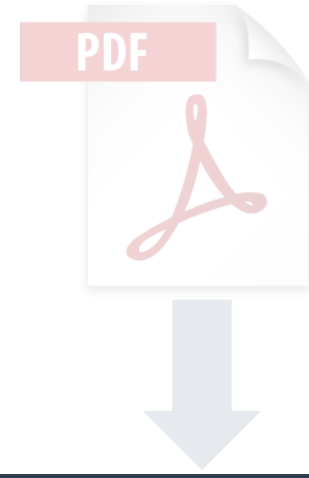
```
processWord(path) {  
    file = openFile(path);  
    rawData = extractDocData(file);  
    data = parseDocData(rawData);  
    structData = validateData(data);  
    persistData(structData);  
    closeFile(file);  
}
```

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?*

FinancialFileCompiler

- + openFile (..)
- + validateData (..)
- + persistData (..)
- + closeFile (..)
- + extractDocData (..)
- + extractExcelData (..)
- + extractPdfData (..)
- + parseDocData (..)
- + parseExcelData (..)
- + parsePdfData (..)
- + processDoc (..)
- + processExcel (..)
- + processPdf (..)

*kako izgleda klasa koja
sadrži ovu funkciju?*

PDF



```
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

FinancialFileCompiler

```
+ openFile (..)
+ validateData (..)
+ persistData (..)
+ closeFile (..)
+ extractDocData (..)
+ extractExcelData (..)
+ extractPdfData (..)
+ parseDocData (..)
+ parseExcelData (..)
+ parsePdfData (..)
+ processDoc (..)
+ processExcel (..)
+ processPdf (..)
```

PDF



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

Dizajn šabloni

Strategy

State

Observer

FinancialFileCompiler

- + openFile (..)
- + validateData (..)
- + persistData (..)
- + closeFile (..)
- + extractDocData (..)
- + extractExcelData (..)
- + extractPdfData (..)
- + parseDocData (..)
- + parseExcelData (..)
- + parsePdfData (..)
- + processDoc (..)
- + processExcel (..)
- + processPdf (..)

Nepromenljivo

Promenljivo

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

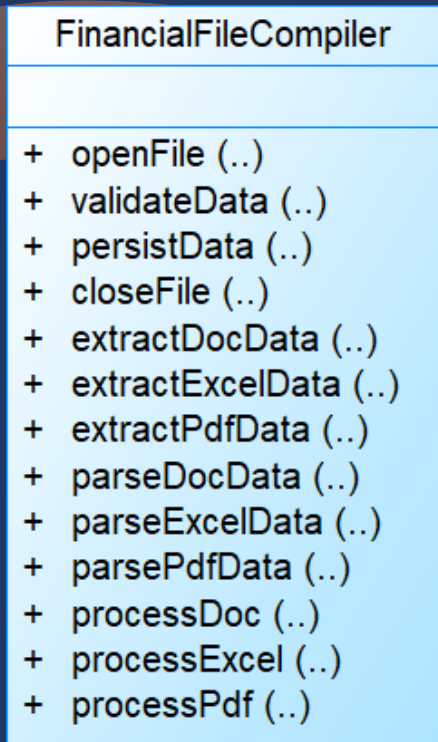
PDF



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

FinancialFileCompiler
{abstract}

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

***final** funkcija daje
strukturu algoritma*

*šta ako nećemo
sve datoteke da
validiramo?*

FinancialDocCompiler

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

FinancialExcelCompiler

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

FinancialPDFCompiler

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

FinancialCsvCompiler

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

*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.

AutoSave Off F1a Project Description Part A - CCaDET v2.1 - Compatibility Mode Nikola Luburic

File Home Insert Design Layout References Mailings Review View Help Search

Clipboard Font Paragraph Styles Editing Voice

1.2. Concept and methodology

The purpose of Clean CaDET is to **automatically detect code smells** in the developer's code and offer personalized advice on how best to deal with the detected smells, thus training the developer to write code of higher quality. Figure 1 displays the conceptual vision of Clean CaDET and represents a communication diagram illustrating the interaction between the major modules of the tool and the software developer.

```
graph LR
    Developer[Developer] -- "1. Write code" --> IDE[IDE]
    IDE <--> C2Plugin((C2Plugin))
    C2Plugin -- "2. Send code representation" --> C2Application
    subgraph C2Application [C2Application]
        direction TB
        CCA([Clean Code Analyzer])
        CCA --> DSA[Dataset of software artifacts]
        DSA -- "3. Send discovered code smells" --> ST([Smart Tutor])
    end
    ST -- "4. Send" --> C2Plugin
    C2Plugin -- "5. Learn" --> Developer
```

Figure 1 illustrates the conceptual vision of Clean CaDET, showing the interaction between the major modules of the tool and the software developer. The diagram depicts a cycle where the Developer writes code in the IDE, which is then processed by the C2Plugin. The C2Plugin sends code representation to the C2Application (Clean Code Analyzer), which uses AI-based smell detection techniques and a Dataset of software artifacts to discover code smells. These smells are then sent to the Smart Tutor, which provides feedback to the C2Plugin, which in turn sends this feedback to the Developer for learning.

Page 3 of 21 4 of 15108 words English (United States) 180%

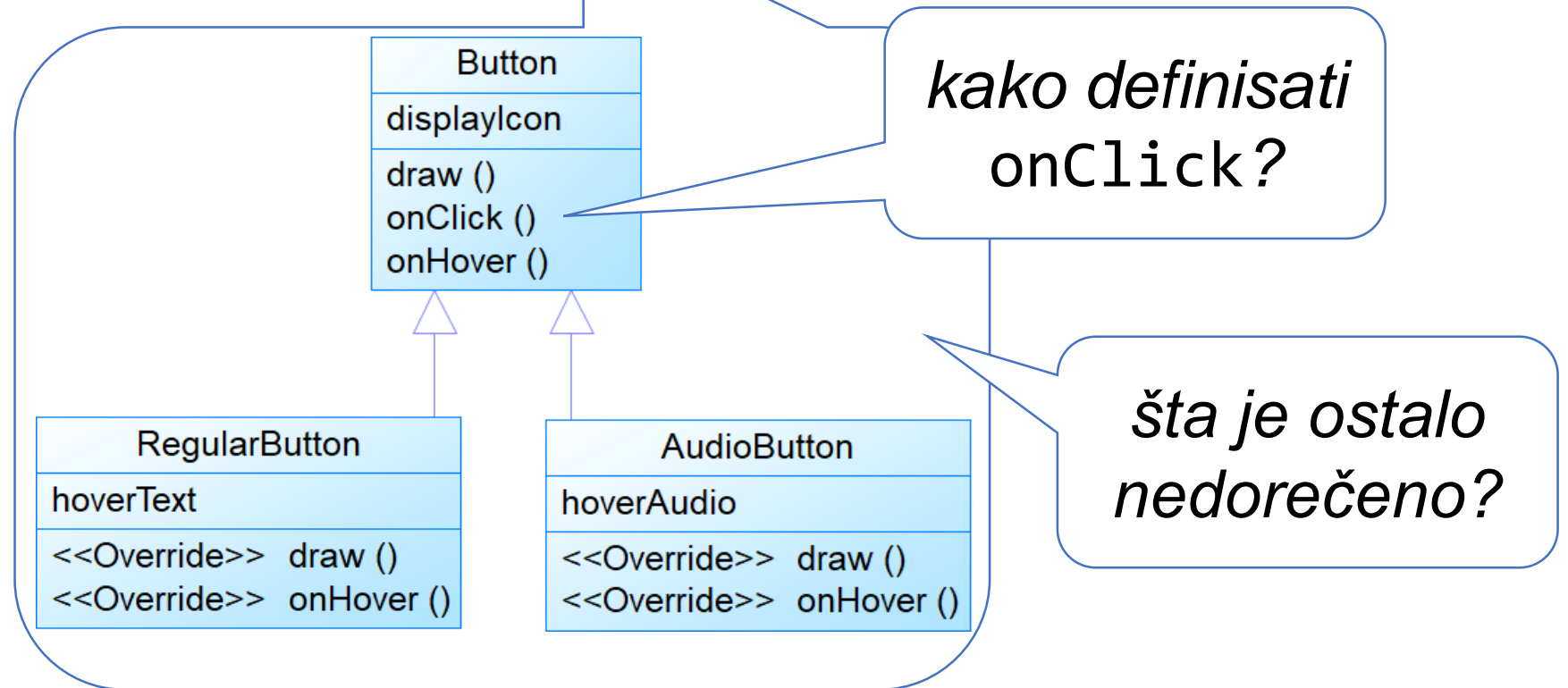
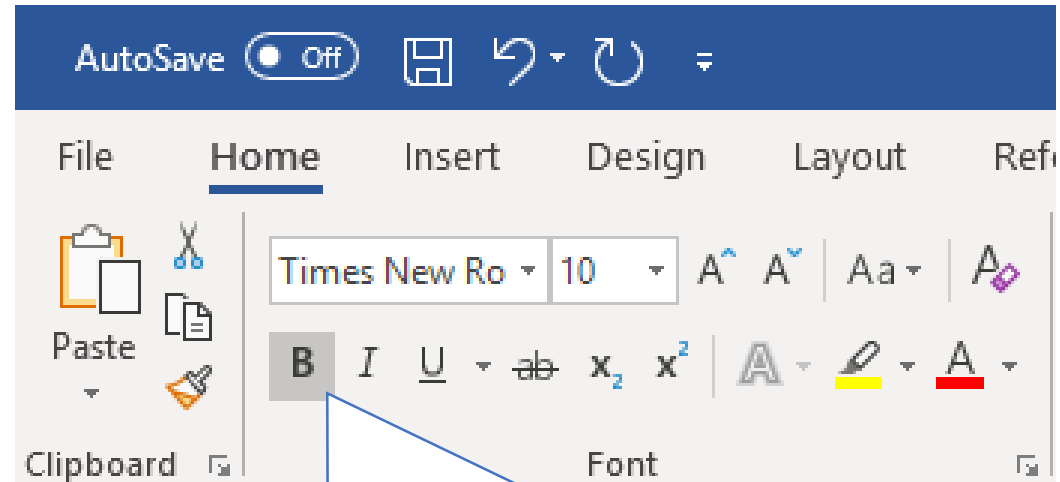
Dizajn šabloni

Strategy

State

Observer

Template M.



Dizajn šabloni

Strategy

State

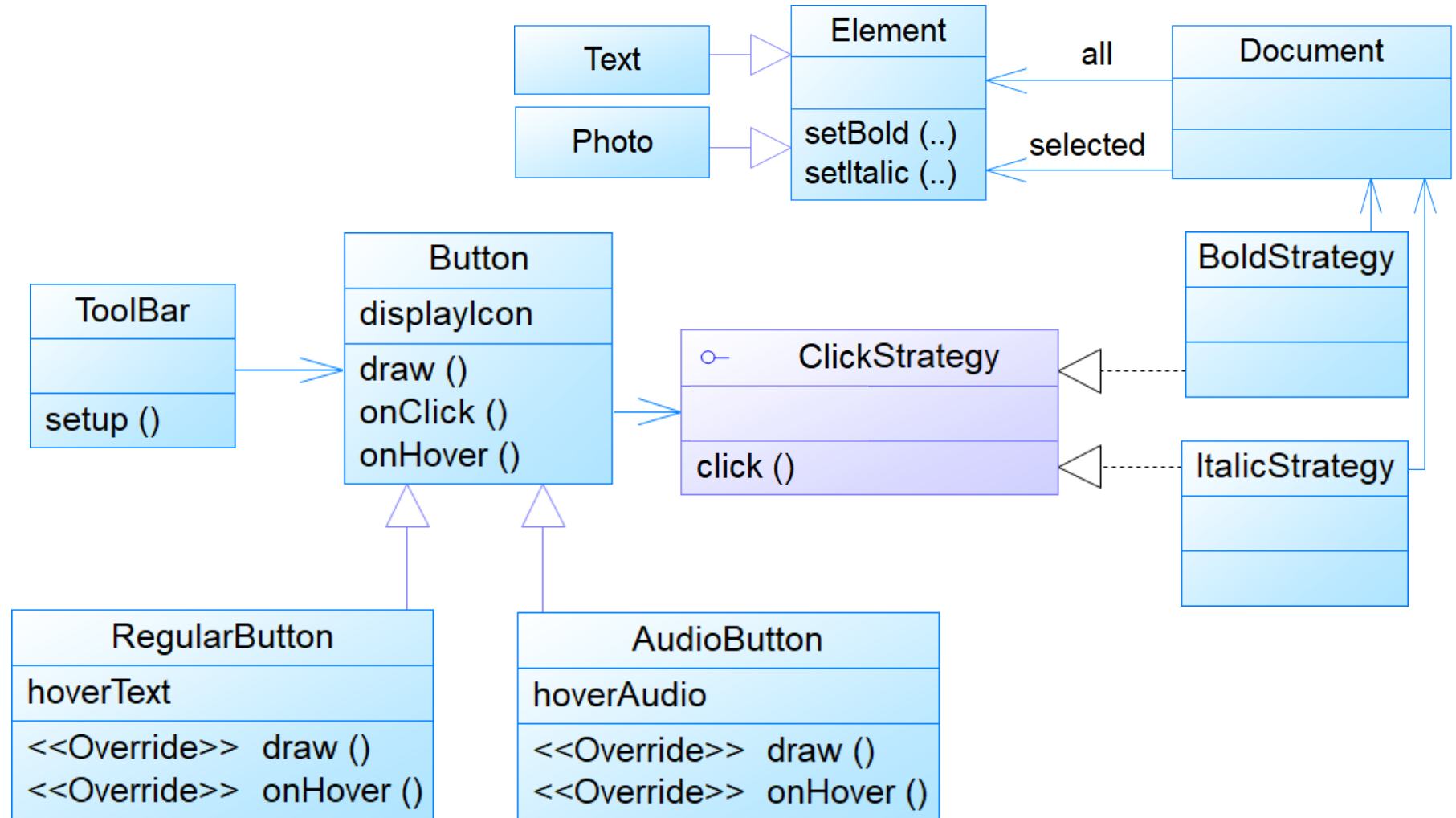
Observer

Template M.

Napiši pseudokod funkcija i dopuni dijagram:

❖ Toolbar > setup() – za **Bold** i *Italic*

❖ BoldStrategy > click()



Dizajn šabloni

Strategy

State

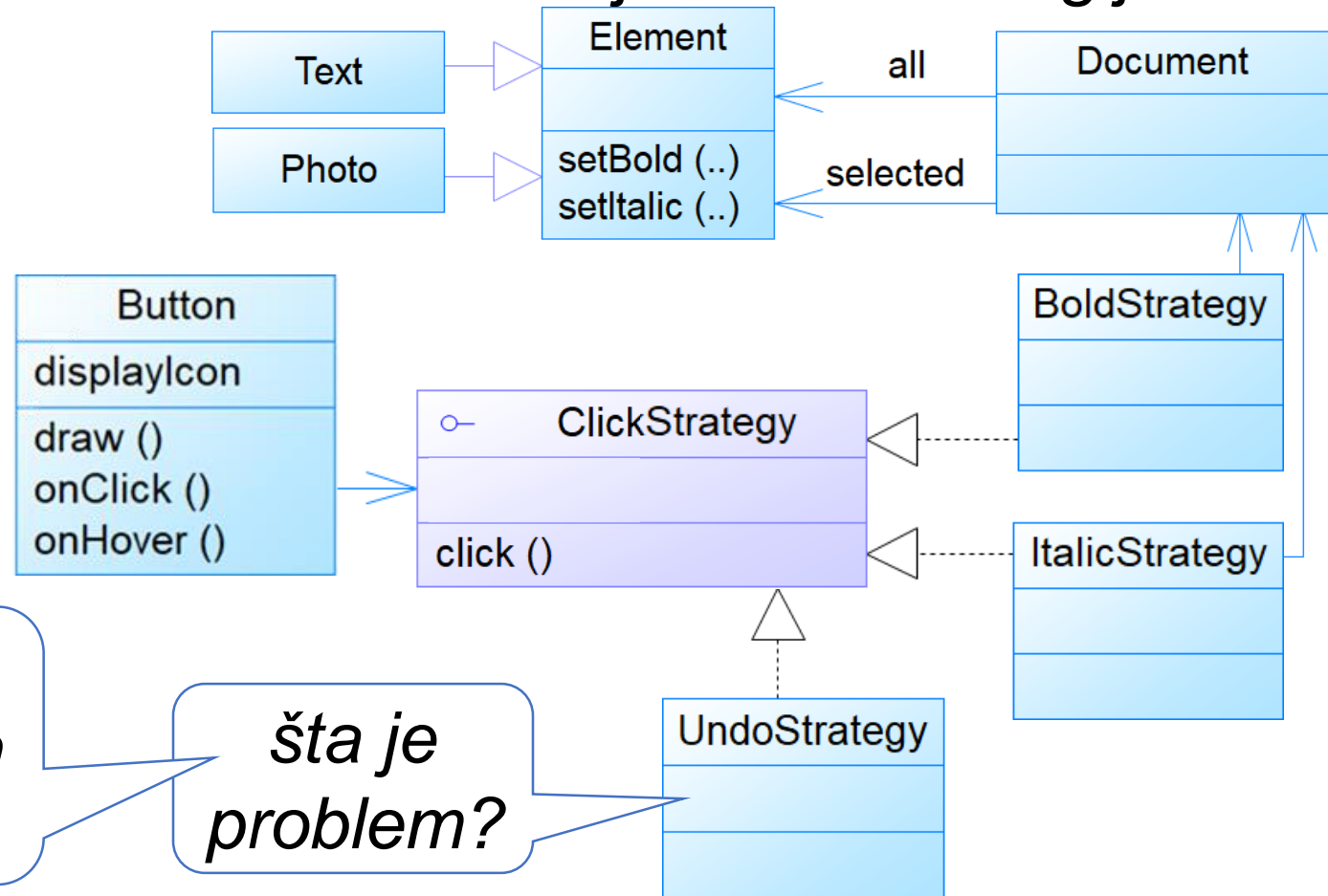
Observer

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

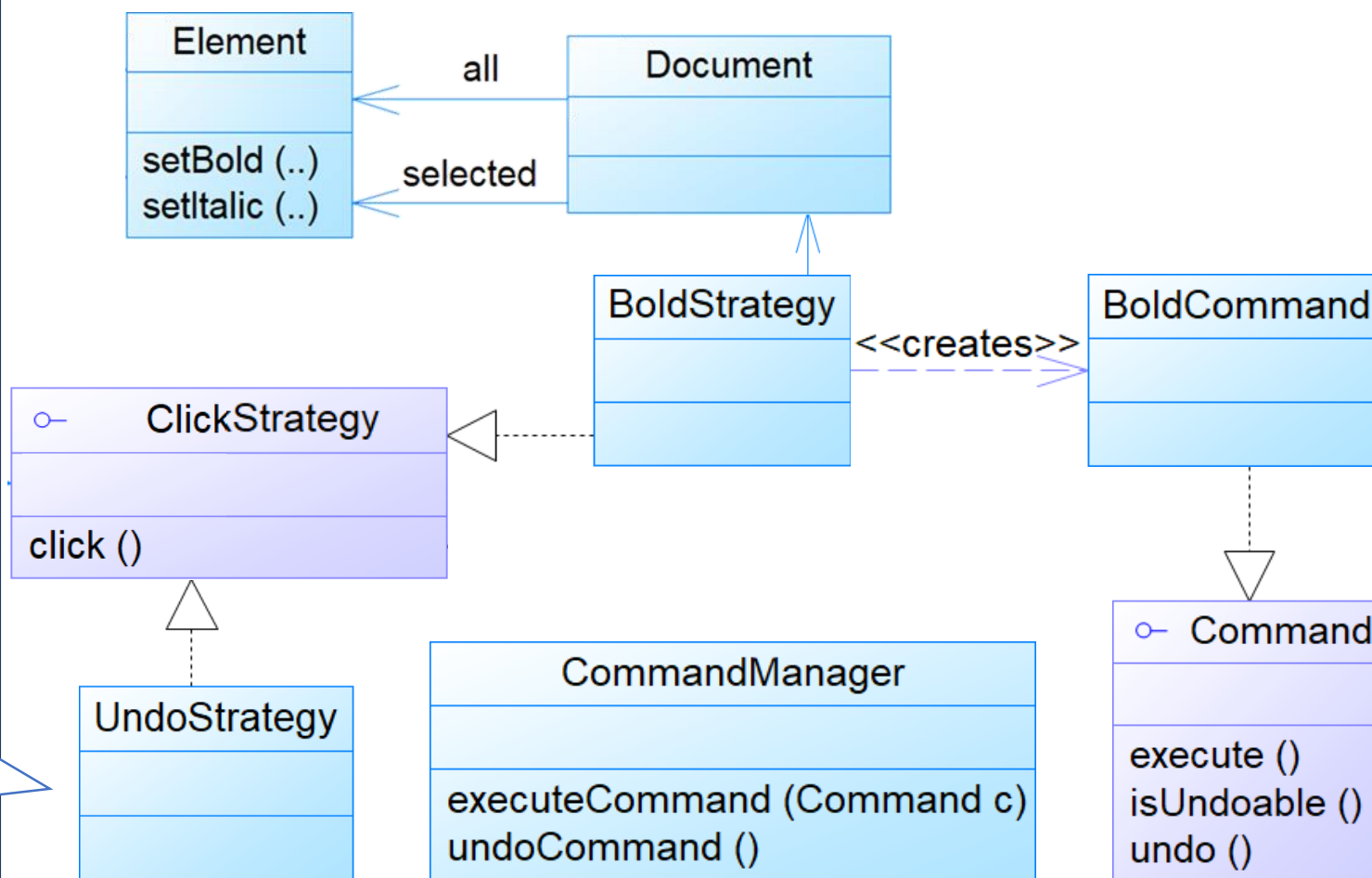
State

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

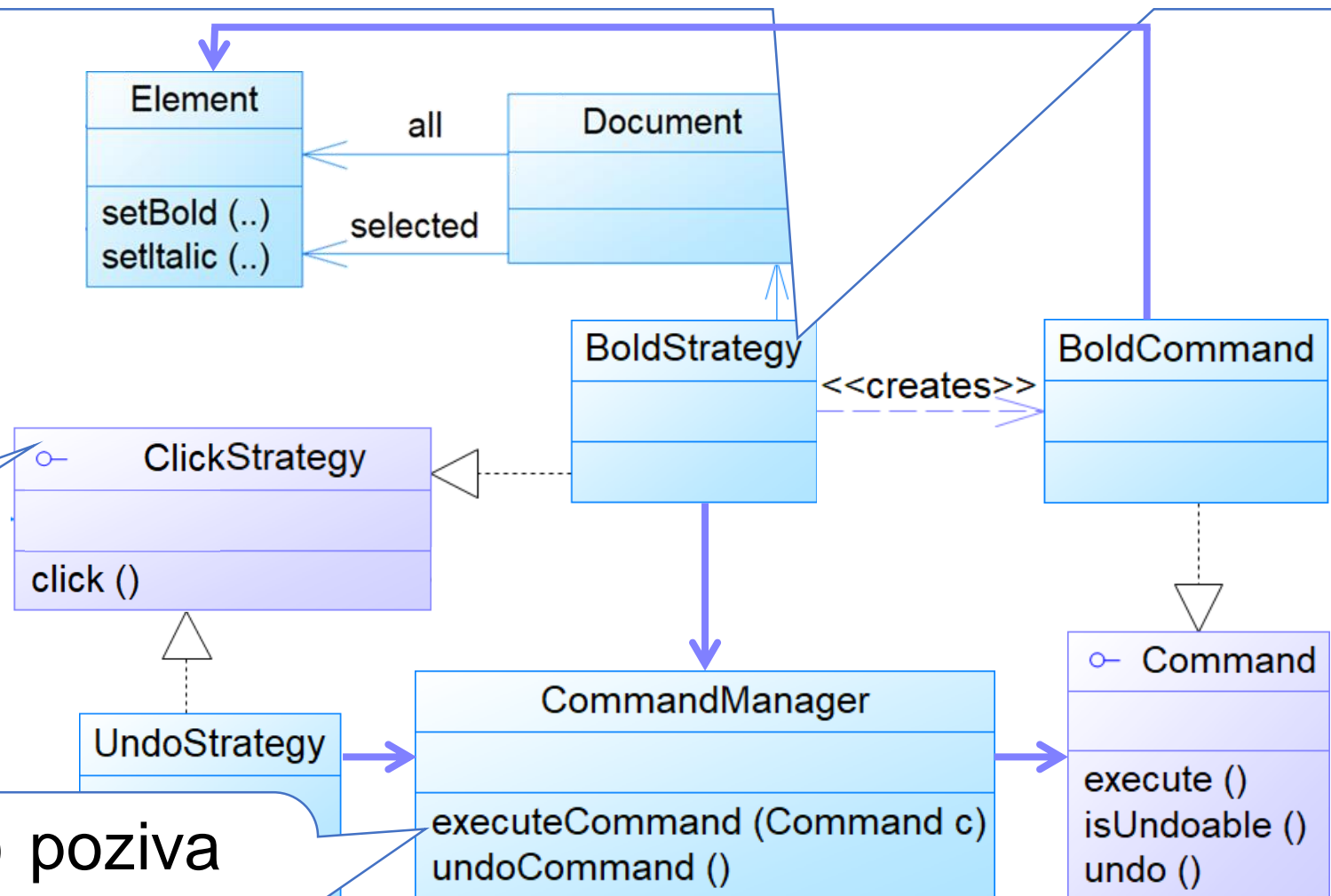
Strategy

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

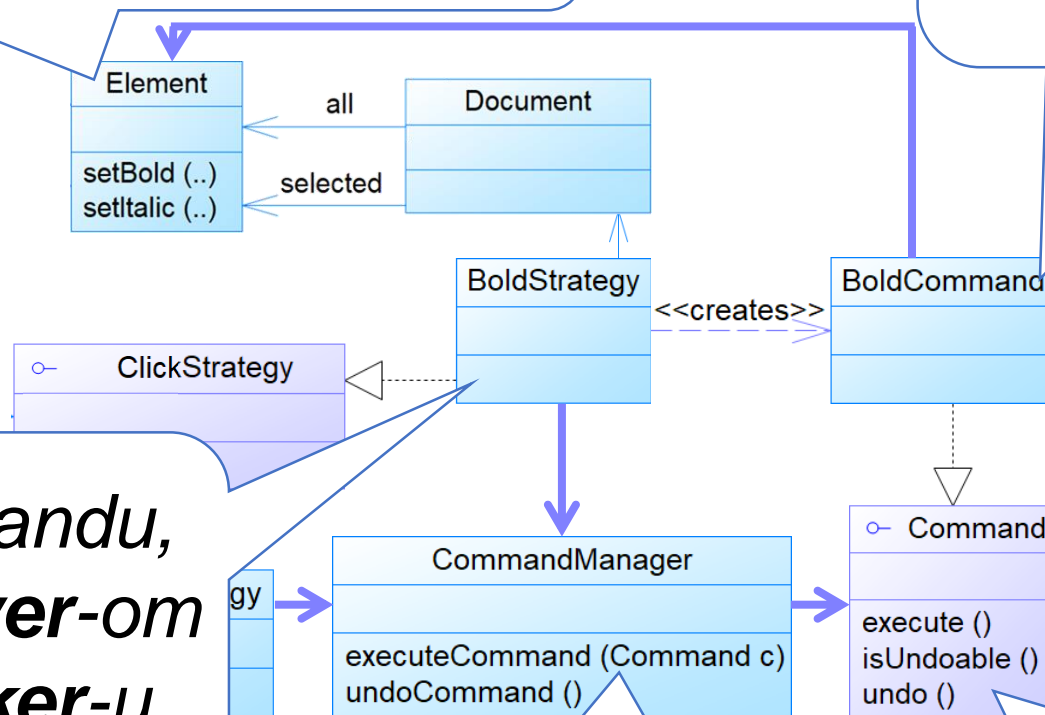
Receiver je objekat koji komanda afektuje

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

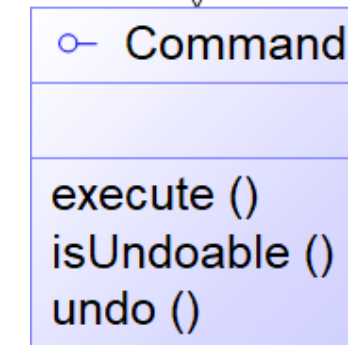
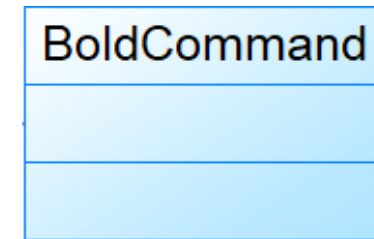
Observer

Template M.

Command Pattern

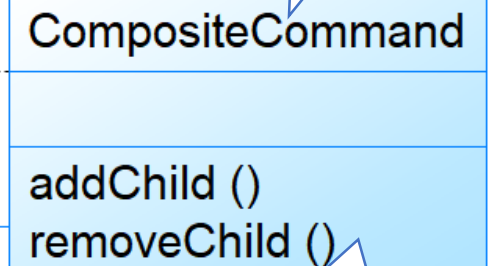
CompositeCommand
InsertLogo

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



*makro
komande*

*šta ovo
povlači?*



*kako izgleda
execute?*

Dizajn šabloni

Strategy

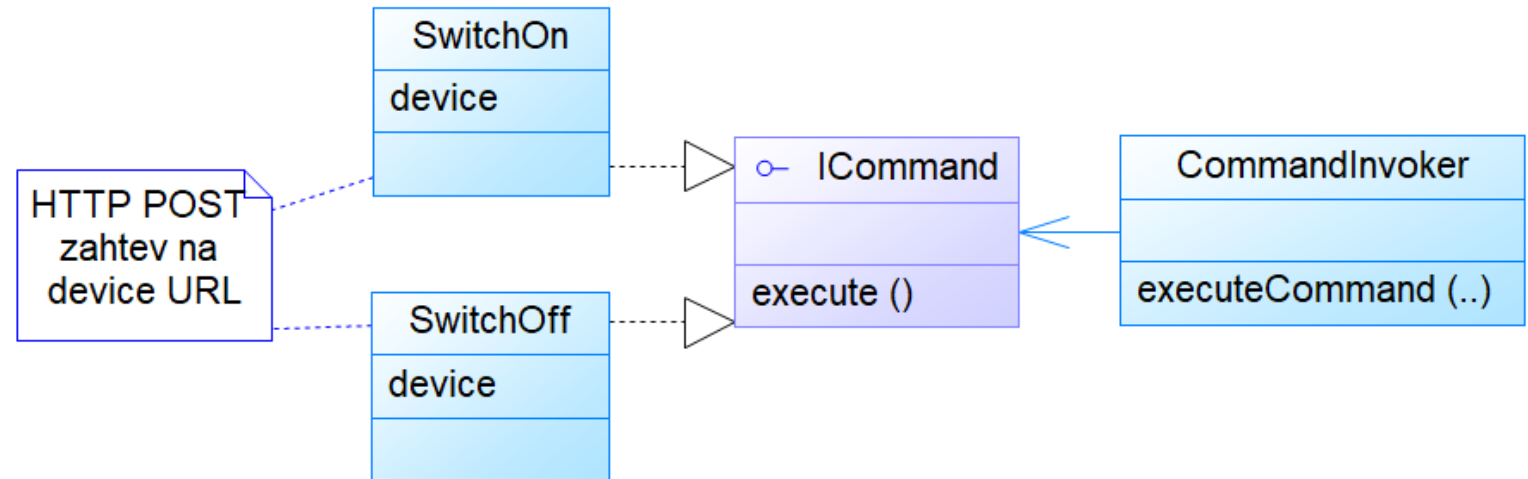
State

Observer

Template M.

Command

Operateri Elektrovojvodine izdaju komande ka trafostanicama da manipuliše energetsom 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 manipulirati