Evaluation Warning: The document was created with Spire.Doc for JAVA.

252

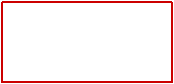
OOAD

GRASP

□ General Responsibility Assignment Software Principles/Patterns -

Design principles

Evaluation Warning : The document was created with Spire.PDF for java.



253

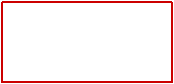
OOAD

Martin Fowler

object-oriented design.

Understanding responsibilities is key to

Evaluation Warning : The document was created with Spire.PDF for java.



254

OOAD

collaborating responsible objects.

□ RDD leads to viewing an OO design as a community of

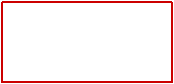
who collaborate with other people to get work done.

□ Think of software objects similar to people with responsibilities

□ RDD is a metaphor for thinking about object-oriented design.

Responsibilities-Driven Design

Evaluation Warning : The document was created with Spire.PDF for java.



255

OOAD

patterns of assigning responsibility.

□ A collection of patterns/principles for achieving good design -

□ A learning aid for OO Design with responsibilities.

new context

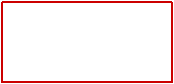
■ Pattern is a solution which can be applied to a problem in a

Principles (GRASP)

□ General Responsibility Assignment Software Patterns or

GRASP

Evaluation Warning : The document was created with Spire.PDF for java.



256

OOAD

□ The object has data that it can calculate or derive

□ The object knows the objects to which it is linked

□ The object knows private encapsulated data

■ Know

□ The object controls or coordinates activities of other objects

□ The object initiates an action of another object

□ The object accomplishes something itself

■ Do

□ Two main types of responsibility

□ The responsibilities of an object relate to the behaviour of an object

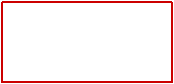
based on its responsibilities

□ The determination of the attributes and operations of a class is essentially

□ A responsibility is an duty or a contract of a class

Responsibility

Evaluation Warning : The document was created with Spire.PDF for java.



257

OOAD

method.

□ Responsibility “create a Sale” can be translated by only one

■ A responsibility can be translated by one method

several methods of several classes

□ Responsibility “offer access to the database” can be translated to

classes

■ A responsibility can be translated by several methods of several

granularity of the responsibilities

□ The traduction of responsibilities into methods of classes depends on the

□ An object of Sale class is responsible for knowing its total (know).

Payment class (do)

□ An object of Sale class is responsible for creating an object of

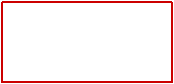
■ Example

phase

□ The responsibilities are assigned to classes during the design

Responsibility

Evaluation Warning : The document was created with Spire.PDF for java.



258

OOAD

:Payment

<<create>>

pay(amount)

:Sale

interaction diagrams

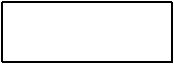
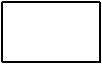
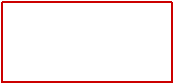
□ The discovery of responsibilities is achieve when building

oriented design.

□ The assignment of responsibilities to objects is very important in object-

Assignment and discovery of responsibilities

Evaluation Warning : The document was created with Spire.PDF for java.



259

OOAD

■ Protected variations

■ Pure fabrication

■ Indirection

■ Polymorphism

system event messages

■ Controller: assigning the responsibility for management of the

responsibilities to classes

■ Information Expert: the common principle when assigning

object

■ Creator: assigning the creation responsibility of an object to another

remains high

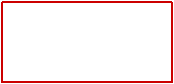
■ High Cohesion: assigning the responsibilities to ensure that cohesion

■ Low Coupling: assigning responsibilities in a low coupling way

□ We consider 5 among 9 GRASP patterns/principles

GRASP patterns

Evaluation Warning : The document was created with Spire.PDF for java.



260

OOAD

Sub-system 1

Sub-system 2

Cohesion

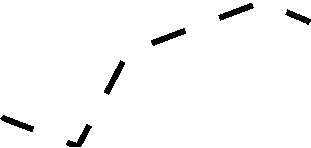
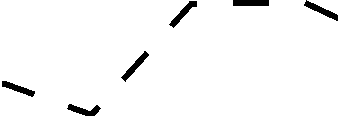
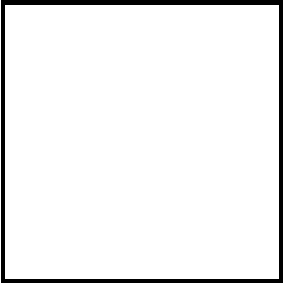
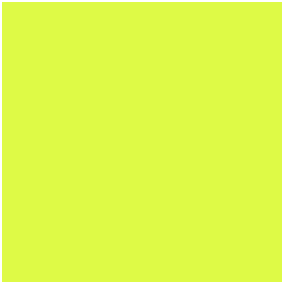
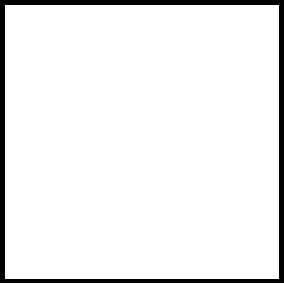
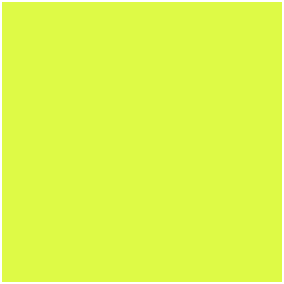
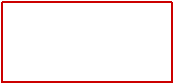
Coupling

□ Cohesion: Amount of relations within sub-system

□ Coupling: Amount of relations between objects/sub-systems

Coupling and Cohesion

Evaluation Warning : The document was created with Spire.PDF for java.



261

OOAD

■ High cohesion makes a module easier to understand

■ Goal: the content of each module are strongly inter-related

□ Maximises cohesion within modules

■ Low coupling makes future change easier

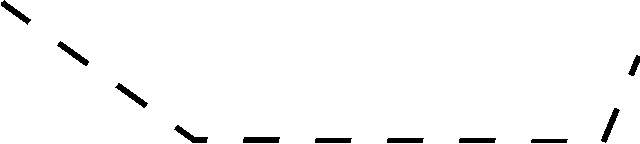
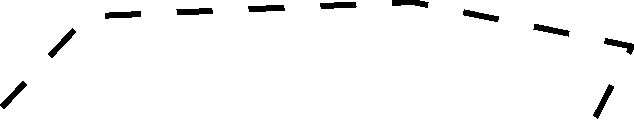
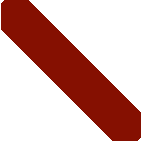
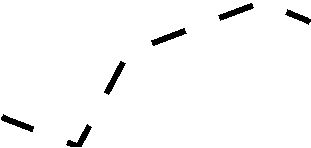
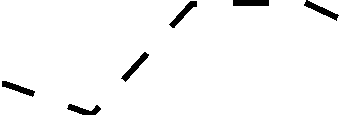
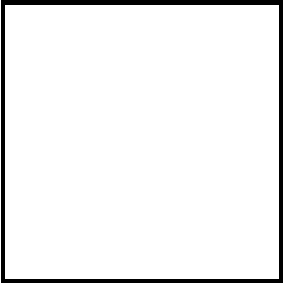
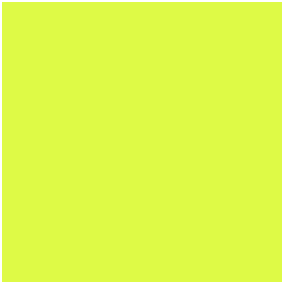
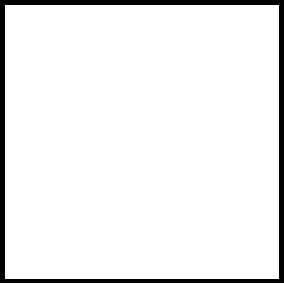
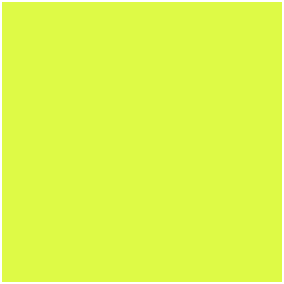
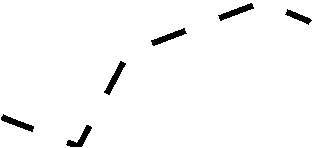
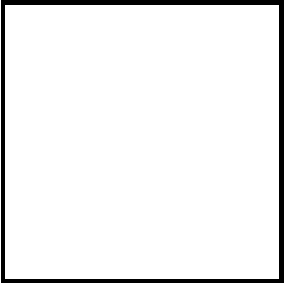
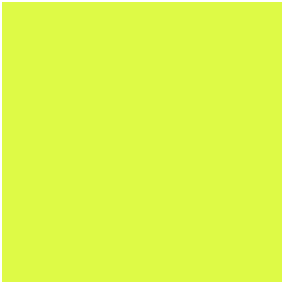
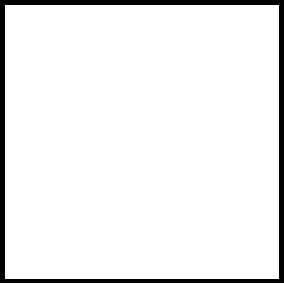
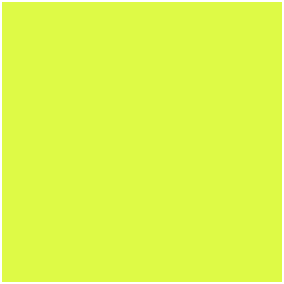
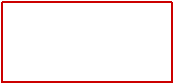
to interact

■ Goal: modules don’t need to know much about one another

□ Minimises coupling between modules

Properties of a good architecture

Evaluation Warning : The document was created with Spire.PDF for java.



262

OOAD

two many other elements

■ An element with low (or weak) coupling is not dependent on

knowledge of or relies on other elements

■ Measure how strongly one element is connected to, has

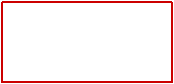
□ Coupling:

and increase reuse?

□ Problem: How to support low dependency, low change impact,

Low coupling

Evaluation Warning : The document was created with Spire.PDF for java.



263

OOAD

■ TypeX is an interface and TypeY implements that interface

■ TypeX is a direct or indirect subclass of TypeY

(parameter, local variable, return type)

■ TypeX has a method that references an instance of TypeY

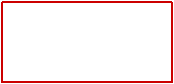
■ A TypeX object calls on services of TypeY object

■ TypeX has an attribute that refers to a TypeY instance

□ Common forms of coupling from TypeX to TypeY

When are two classes coupled?

Evaluation Warning : The document was created with Spire.PDF for java.



264

OOAD

presence of the classes on which it is dependent

■ Harder to reuse because its use requires the additional

■ Harder to understand in isolation

■ Force local changes because of changes in related classes

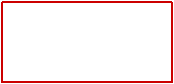
following problems:

classes. Such classes may be undesirable and suffer from the

□ A class with high (or strong) coupling relies on many other

High coupling (Bad)

Evaluation Warning : The document was created with Spire.PDF for java.



265

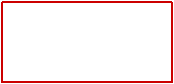
OOAD

□ Use this principle to evaluate alternatives

□ Assign responsibility so that coupling remain low

Solution

Evaluation Warning : The document was created with Spire.PDF for java.



266

OOAD

□ How can we assign responsibilities to adhere to Low Coupling pattern?

associate it with Sale.

□ Supposing that we would like to create an instance of Payment and

Register

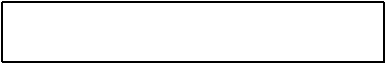
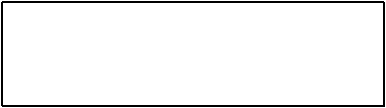
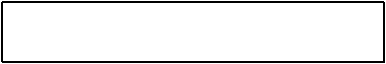
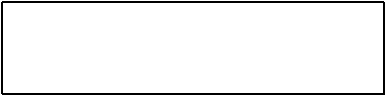
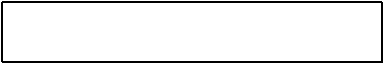
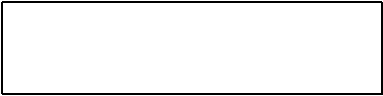
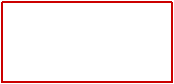
Payment

Sale

□ We have three following class in the Cash Register system

Example

Evaluation Warning : The document was created with Spire.PDF for java.



267

OOAD

:Sale

2: addPayment(p)

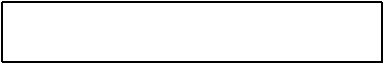
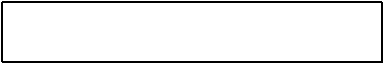
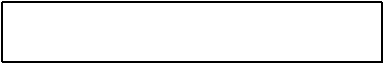
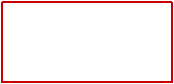
:Register

p:Payment

1: create()

Solution 1

Evaluation Warning : The document was created with Spire.PDF for java.



268

OOAD

:Payment

1: create()

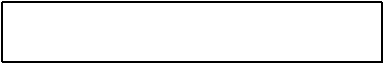
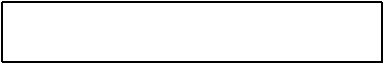
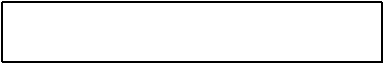
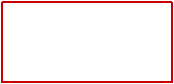
:Register

:Sale

1: createPayment()

Solution 2

Evaluation Warning : The document was created with Spire.PDF for java.



269

OOAD

□ Do not consider patterns in isolation

different solutions

□ Note that two patterns - Low Coupling and Creator - suggest

□ Solution 2 therefore has lower coupling

absent in Solution 2

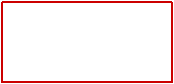
□ Solution 1 has the coupling of Register and Payment, which is

Payment.

□ Assume that each Sale will eventually be coupled with a

Which solution is better?

Evaluation Warning : The document was created with Spire.PDF for java.



270

OOAD

:Payment

1: create()

:Register

:Sale

Lower coupling

1: createPayment()

□ Solution 2: Register and Sale are coupled, Sale and Payment are coupled.

:Sale

2: addPayment(p)

:Register

p:Payment

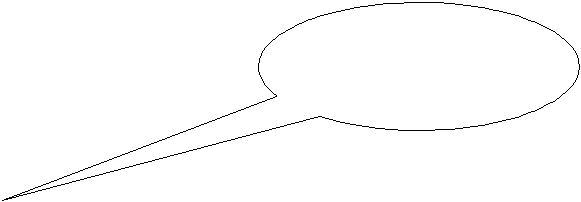
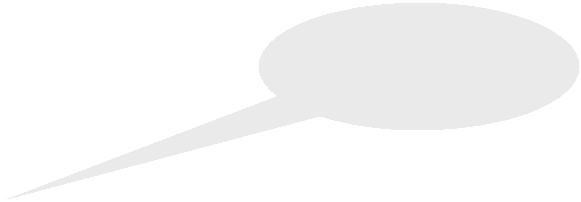
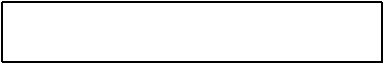
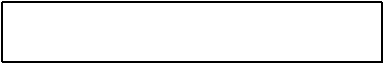
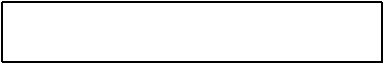
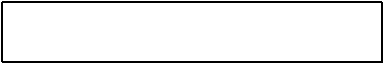
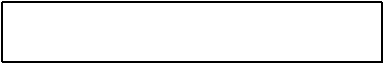
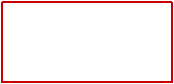
1: create()

both Payment and Sale.

□ Solution 1: Register knows both Payment and Sale. Register depends on

Solutions

Evaluation Warning : The document was created with Spire.PDF for java.



271

OOAD

■ Easily to understand and maintain

□ Benefit

■ Gather related responsibilities into an element

■ Clearly define the purpose of the element

□ Solution

functionally related?

■ How to ensure that the operations of any element are

□ Problem

High Cohesion pattern

Evaluation Warning : The document was created with Spire.PDF for java.



272

OOAD

■ constantly affected by change

■ hard to maintain

■ hard to reuse

■ hard to comprehend

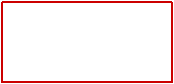
the following problems:

too much work. Such classes are undesirable; they suffer from

□ A class with low cohesion does many unrelated things or does

Low cohesion

Evaluation Warning : The document was created with Spire.PDF for java.



273

OOAD

insertDB()

accessDB(), dbCalls()

getStudentDetails()

insertDB()

dbCalls()

accessDB()

:Student

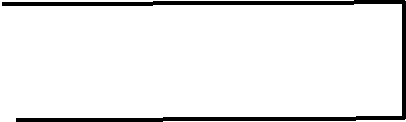
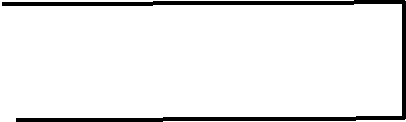
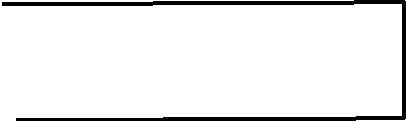
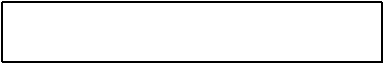
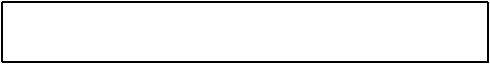
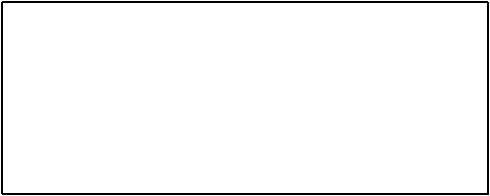
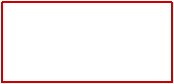
getStudentDetails()

Student

□ Example for Low Cohesion

High Cohesion pattern

Evaluation Warning : The document was created with Spire.PDF for java.



274

OOAD

insertDB(data)

accessDB(), dbCalls()

insertStudent(data)

insertDB()

dbCalls()

accessDB()

DB

getStudentDetails()

:Student

:DB

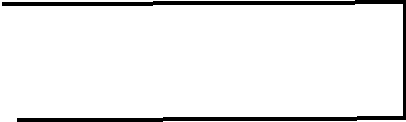
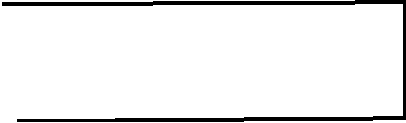
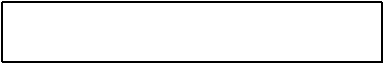
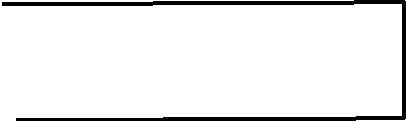
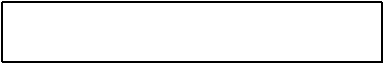
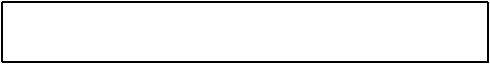
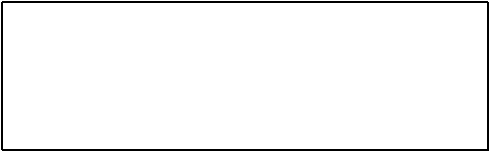
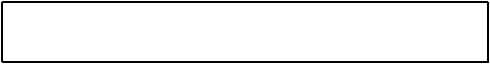
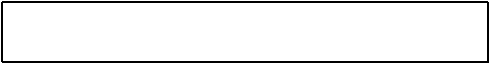
getStudentDetails()

Student

□ Example for High Cohesion

High Cohesion pattern

Evaluation Warning : The document was created with Spire.PDF for java.



275

OOAD

■ have high relatedness of code

■ not do too much work

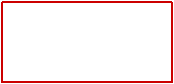
■ have a small number of lines of code

■ have few methods

□ For high cohesion, a class must

Rules of thumb

Evaluation Warning : The document was created with Spire.PDF for java.



276

OOAD

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

time

Sale

■ Who should be responsible for creating a SalesLineItem instance?

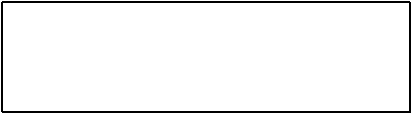
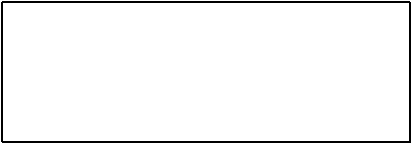
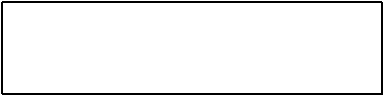
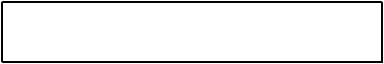
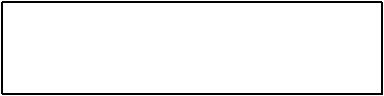
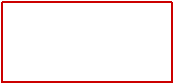
□ Example

■ Who is responsible for creating objects/instances of a class?

□ Problem

“creator” pattern

Evaluation Warning : The document was created with Spire.PDF for java.



277

OOAD

□ “makeLineItem(quantity)” method will be introduced to Sale class

:SalesLineItem

create(quantity)

makeLineItem(quantity)

:Register

:Sale

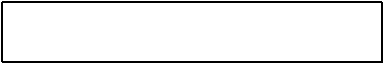
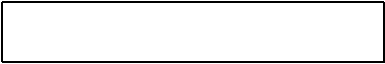
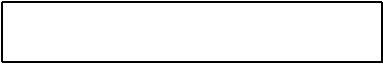
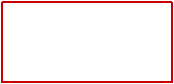
creating objects of SalesLineItem

■ Sale contains SalesLineItem, so Sale should be responsible for

□ Example (continue)

“creator” pattern

Evaluation Warning : The document was created with Spire.PDF for java.



278

OOAD

□ B closely uses A

□ A has the initialising data for A

□ B aggregates A

□ B contains A

of these is true

■ Assign class B the responsibility to create an instance of class A if one

in.

is passed into client. e.g., ProductionDescription needs to be passed

■ Also need initialisation data to be nearby - sometimes requires that it

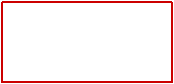
created object in any event

■ Basic idea is to find a creator that needs to be connected to the

□ Discussion

“creator” pattern

Evaluation Warning : The document was created with Spire.PDF for java.



279

OOAD

the “creator” class

□ The coupling is not increased because the created class is visible to

□ Fewer dependencies and more reusability

■ The “creator” pattern supports the low coupling between classes

□ Advantages

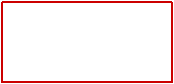
■ Help to find the class who is responsible for creating objects

■ Guide in the assigning responsibility for creating objects

□ Application

“creator” pattern

Evaluation Warning : The document was created with Spire.PDF for java.



280

OOAD

working together to fulfil the responsibility

among several objects or classes, this implies several “partial experts”

■ Accomplishing of a responsibility can request information distributed

■ One of the most used patterns in object-oriented design

□ Application

information to fulfil the responsibility

■ Assign responsibility to the information expert - the class that has the

□ Solution

extend and reuse.

□ Assigning well makes our design easier to understand, maintain,

□ To which ones do we assign a particular functionality?

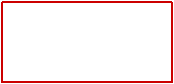
□ Consider that there may be 100s or 1000s of classes

■ What is the general principle of assigning responsibilities to objects?

□ Problem

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



281

OOAD

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

time

Sale

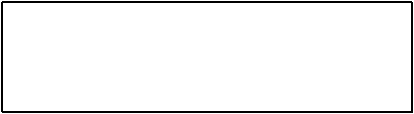
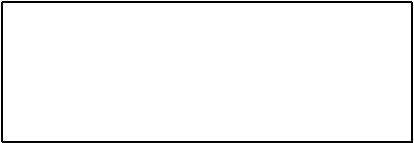
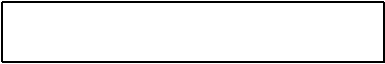
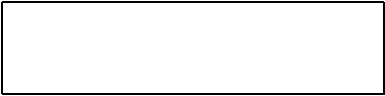
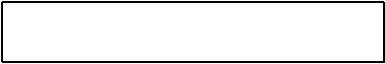
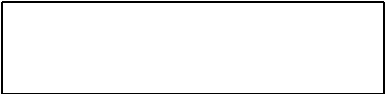
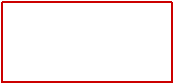
total of a Sale?

■ In the CashRegister system, who is responsible for knowing the grand

□ Example

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



282

OOAD

ProductDescription

knowsproductprice

SaleLineItem

knowslineitemssubtotal

Sale

knowssaletotal

Class

Responsibility

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

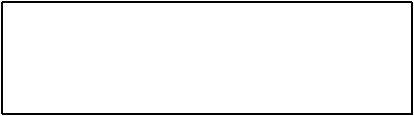
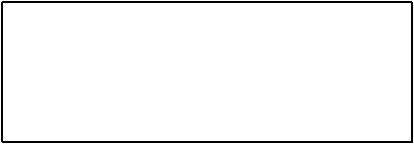
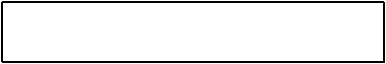
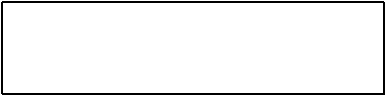
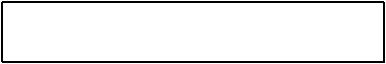
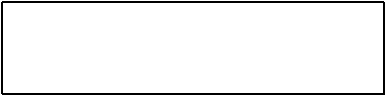
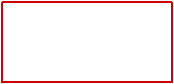
time

Sale

□ Example: Responsibilities

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



283

OOAD

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

time

Sale

■ According to the pattern, Sale knows the information

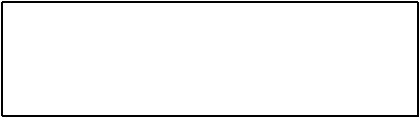
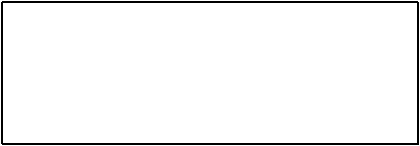
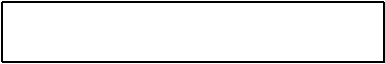
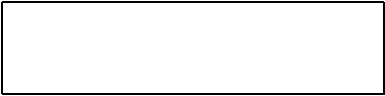
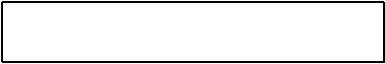
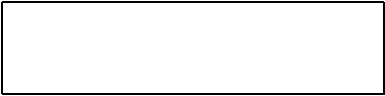
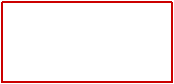
instances of SalesLineItem and the sub-total of each instance.

■ To calculate grand total of a Sale, it is necessary to know the

□ Example (continue)

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



284

OOAD

:Sale

getTotal()

t = getTotal()

time

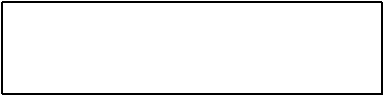
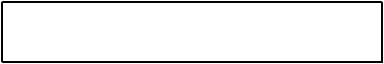
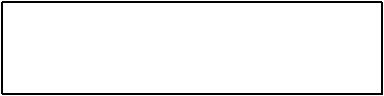
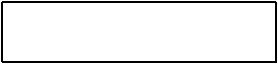
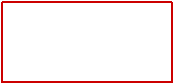
Sale

■ Introduce “getTotal()” method to Sale class

□ Example (continue)

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



285

OOAD

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

time

Sale

■ According to the pattern, SalesLineItem is the expert.

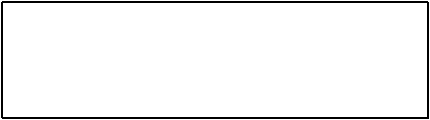
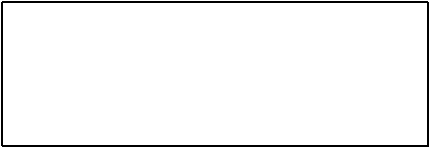
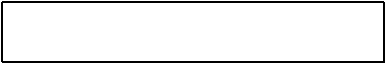
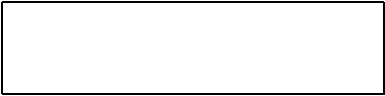
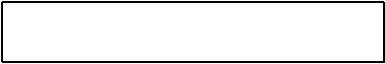
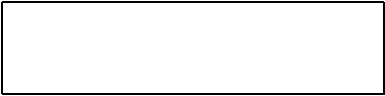
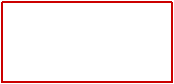
do so, we need to know the number of ProductDescription

■ Then, we need to determine the sub-total of each SalesLineItems. To

□ Example

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



286

OOAD

getSubTotal()

quantity

SalesLineItem

SalesLineItem

:Sale

getTotal()

lineItems[i] :

t = getTotal()

1: \*: st = getSubTotal()

time

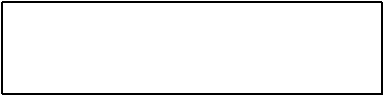
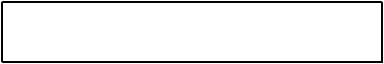
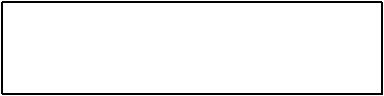
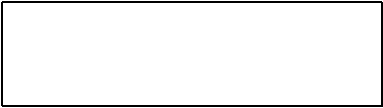
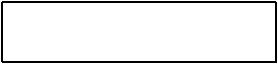
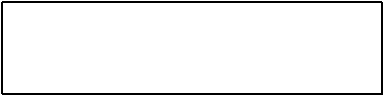
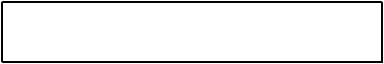
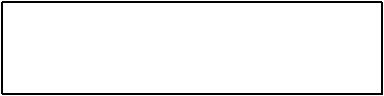
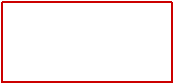
Sale

■ Introduce the “getSubTotal()” method to SalesLineItem class

□ Example

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



287

OOAD

itemID

price

quantity

description

SalesLineItem

ProductDescription

\*

described-by

1

1..\*

contains

1

time

Sale

■ ProductionDescription est expert.

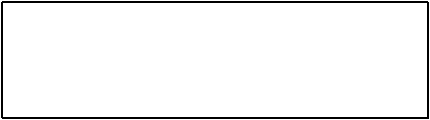
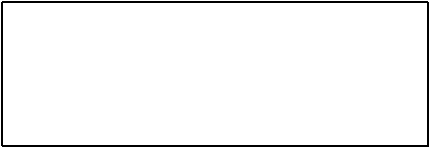
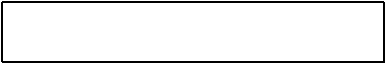
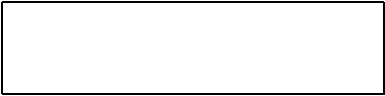
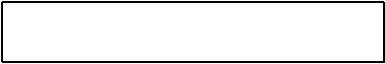
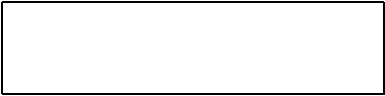
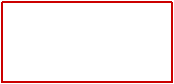
each product.

■ To calculate the sub-total, SalesLineItem needs to know the price of

□ Example

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



288

OOAD

getPrice()

itemID

price

description

ProductDescription

:ProductDescription

getSubTotal()

1.1: getPrice()

quantity

SalesLineItem

SalesLineItem

:Sale

getTotal()

lineItems[i] :

t = getTotal()

1: \*: st = getSubTotal()

time

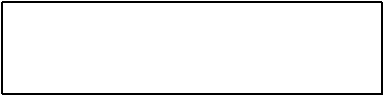
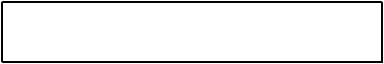
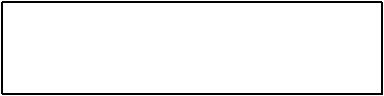
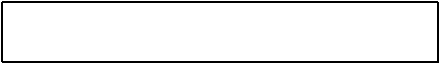
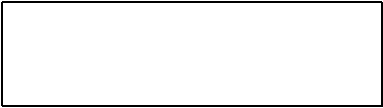
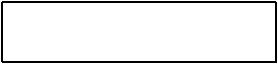
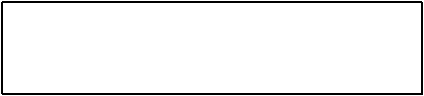
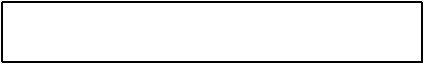
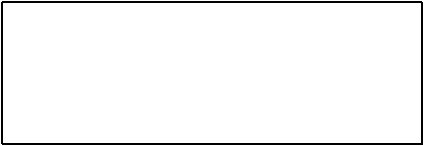
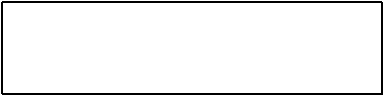
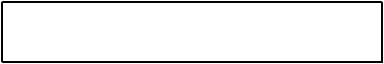
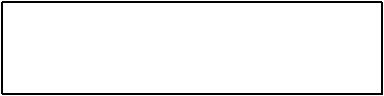
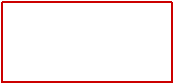
Sale

■ Introduce the “getPrice()” method to ProductDescription class

□ Example

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



289

OOAD

smaller definitions are easier to understand and maintain

the necessary information, it encourages more coherent and

■ The behaviour is distributed among the classes that possess

to be more robust and easier to maintain

■ This pattern supports loose coupling, this allows the system

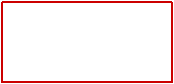
information to satisfy responsibility

■ The encapsulation is maintained since objects use their own

□ Advantages

Information Expert pattern

Evaluation Warning : The document was created with Spire.PDF for java.



290

OOAD

:Class ???

Business Logic Layer

events / requests

Web UI

Desktop UI

Mobile UI

Presentation Layer

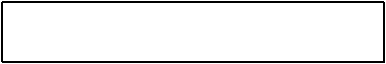
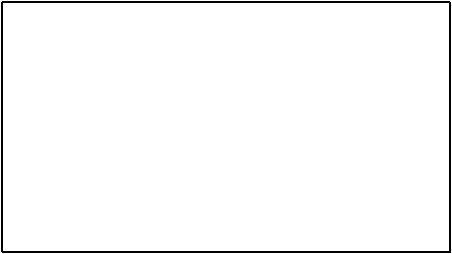
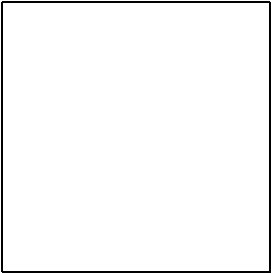
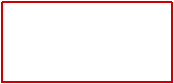
coordinates (“controls”) a system operation?

■ Which first object beyond the User Interface (UI) layer receives and

□ Problem

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



291

OOAD

:Controller

Business Logic Layer

events / requests

Web UI

Desktop UI

Mobile UI

Presentation Layer

only receives the requests but doesn’t not actually solve them.

■ A controller should delegate the work to other objects. The controller

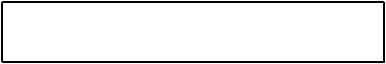
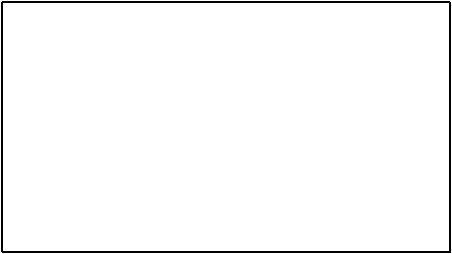
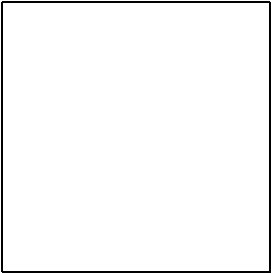
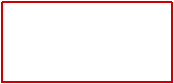
for receiving and handling a system operation.

■ A Controller is the first object beyond the UI layer that is responsible

□ Solution

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



292

OOAD

events)?

■ What class can be the controller (i.e., what class processes the

Desktop UI

….

enterReturnItem()

makeReturnItem()

Presentation Layer

makePayment()

makeNewSale()

enterItem()

endSale()

Web UI

System

■ The Cash Register system has several events

□ Example

■ A controller class is selected to process the events

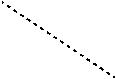
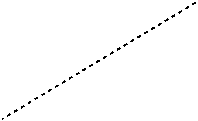
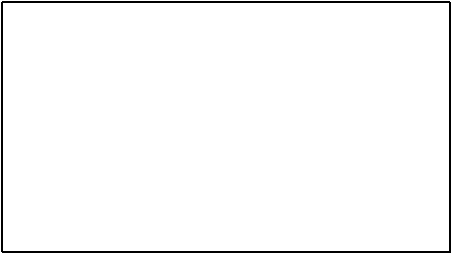
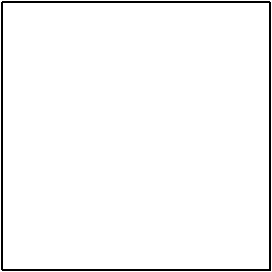
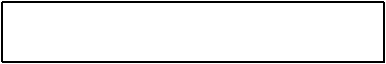
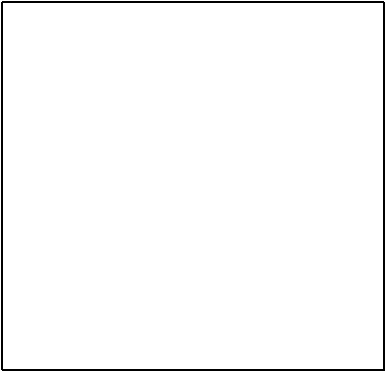
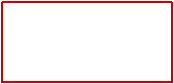
process external events

■ The Controller pattern can be applied to all the systems that need to

□ Application

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



293

OOAD

….

enterReturnItem()

makeReturnItem()

makePayment()

Business Logic Layer

makeNewSale()

enterItem()

endSale()

Register

events / requests

Desktop UI

….

enterReturnItem()

makeReturnItem()

makePayment()

Presentation Layer

makeNewSale()

enterItem()

endSale()

Web UI

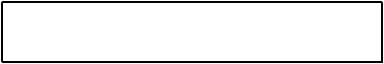
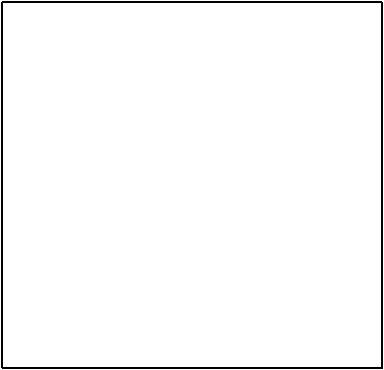
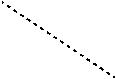
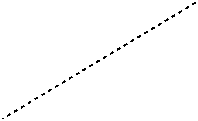
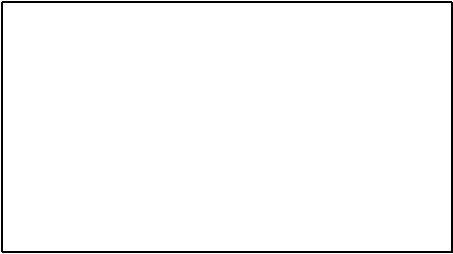
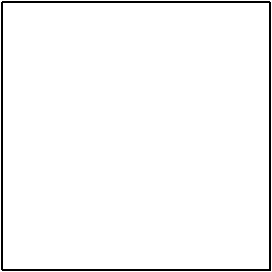
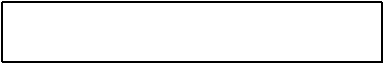
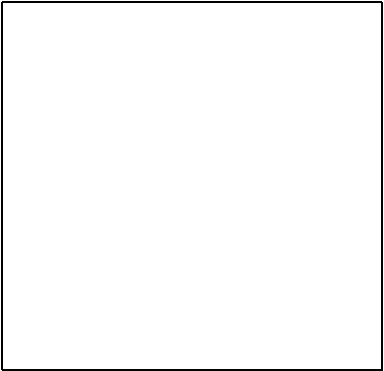
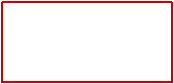
System

■ Solution 1: use one controller

□ Example: Cash Register system

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



294

OOAD

makePayment()

Business Logic Layer

makeNewSale()

….

enterItem()

enterReturnItem()

endSale()

makeReturnItem()

ProcessSaleHandler

HandleReturnsHandler

events / requests

Desktop UI

….

enterReturnItem()

makeReturnItem()

makePayment()

Presentation Layer

makeNewSale()

enterItem()

endSale()

Web UI

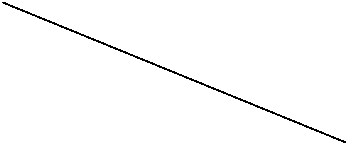
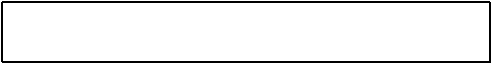
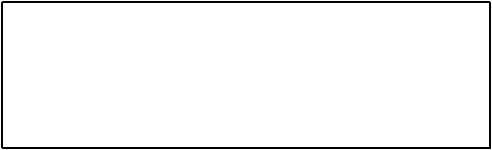
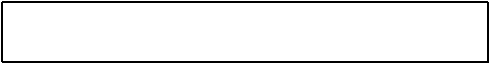
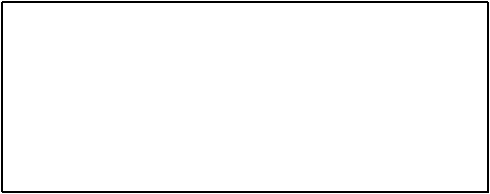
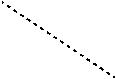
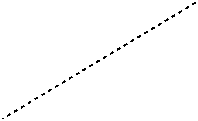
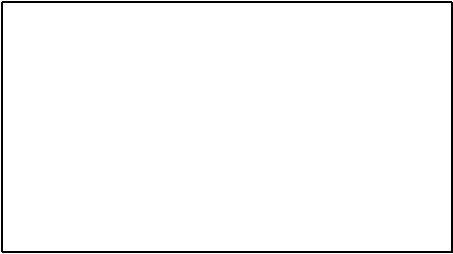
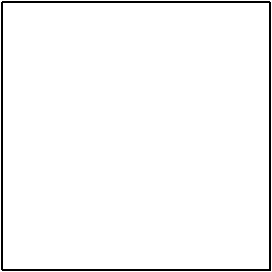
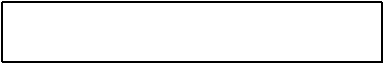
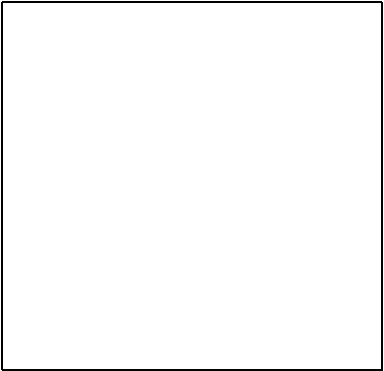
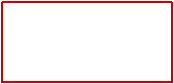
System

■ Solution 2: use several controllers

□ Example: Cash Register system

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



295

OOAD

:Controller

Business Logic Layer

events / requests

Web UI

Desktop UI

Mobile UI

Presentation Layer

example, to ensure that operations occur in a legal sequence.

□ Creates opportunity to reason about state of a use-case, for

□ Increase potential for reuse and pluggable interfaces

application logic

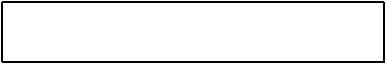
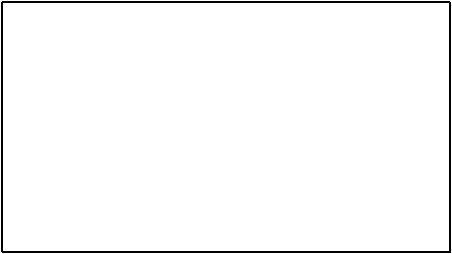
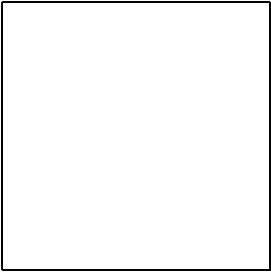
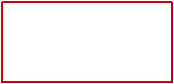
□ This is simply a delegation pattern - the UI should not contain

■ Advantages

□ Discussion

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.



296

OOAD

makePayment()

Business Logic Layer

makeNewSale()

….

enterItem()

enterReturnItem()

endSale()

makeReturnItem()

ProcessSaleHandler

HandleReturnsHandler

events / requests

…

Presentation Layer

System

each system operation to other objects.

■ Design controller so that it primarily delegates the fulfilment of

■ Add more controllers

□ Remedies

information found elsewhere), etc.

the work handling events, has to many attribute (duplicating

□ a single controller that receives all system events, does too much of

■ Difficulty: Bloated controllers

□ Discussion

Controller pattern

Evaluation Warning : The document was created with Spire.PDF for java.

