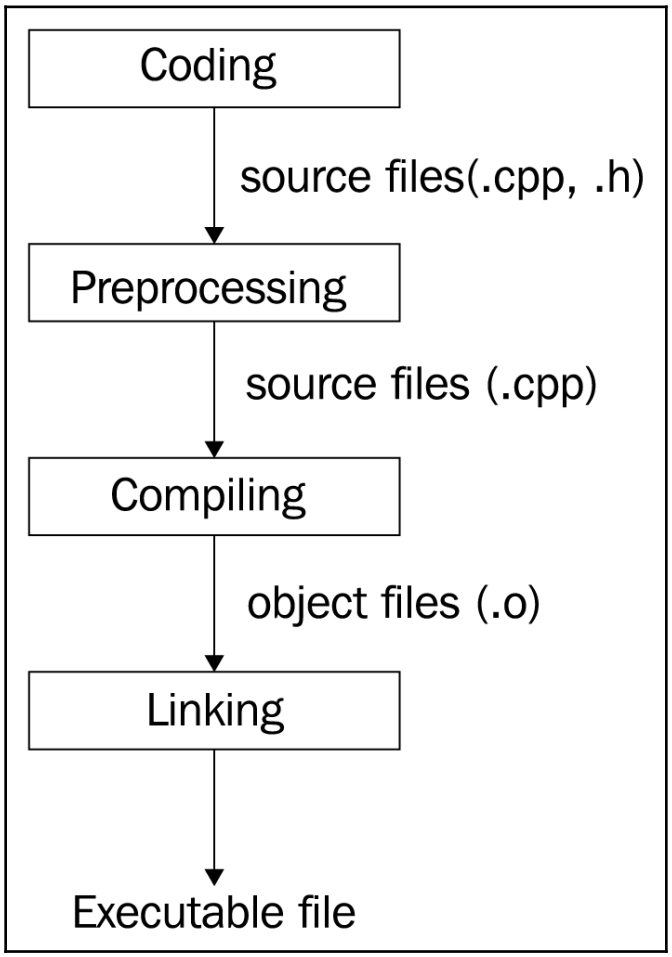
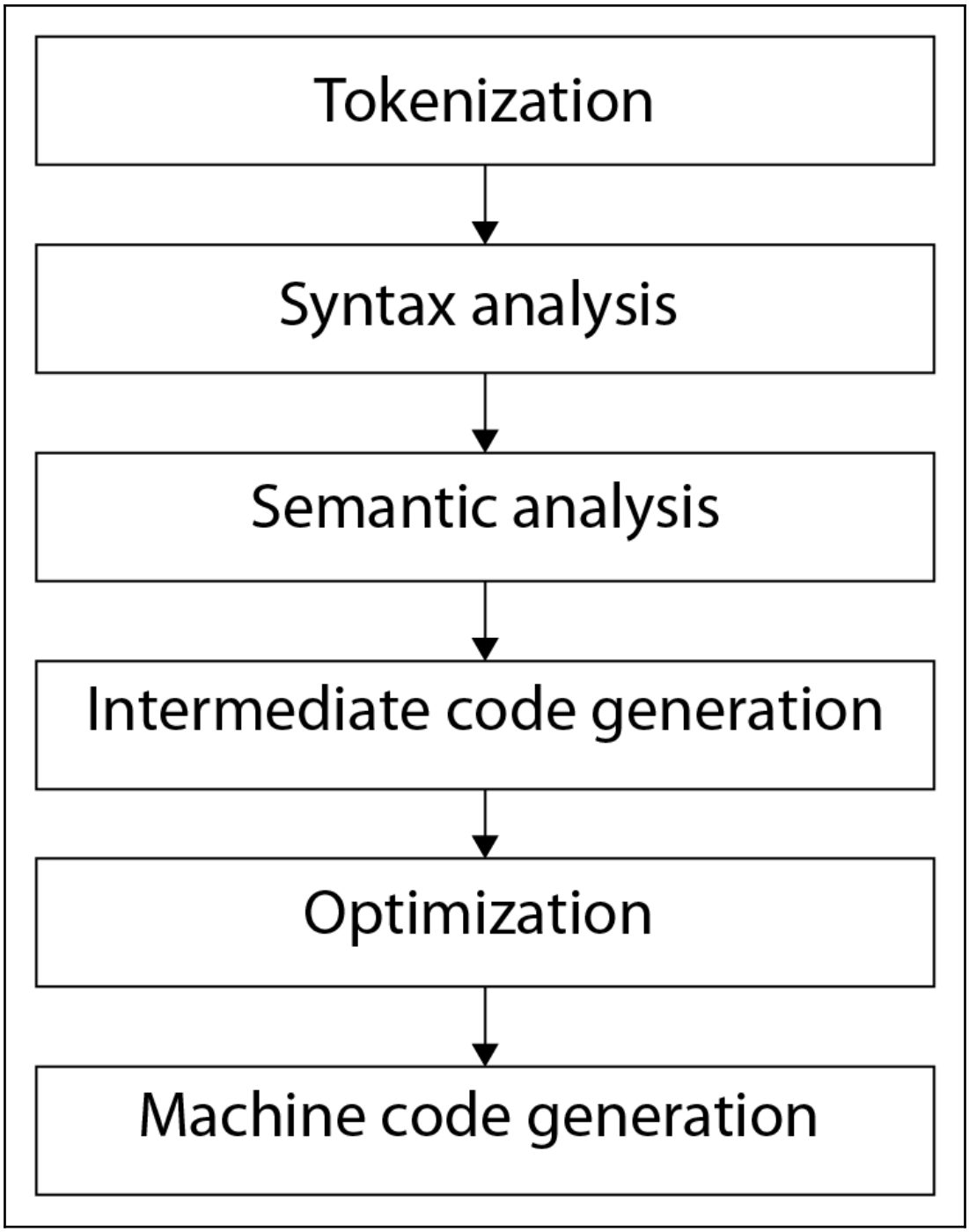
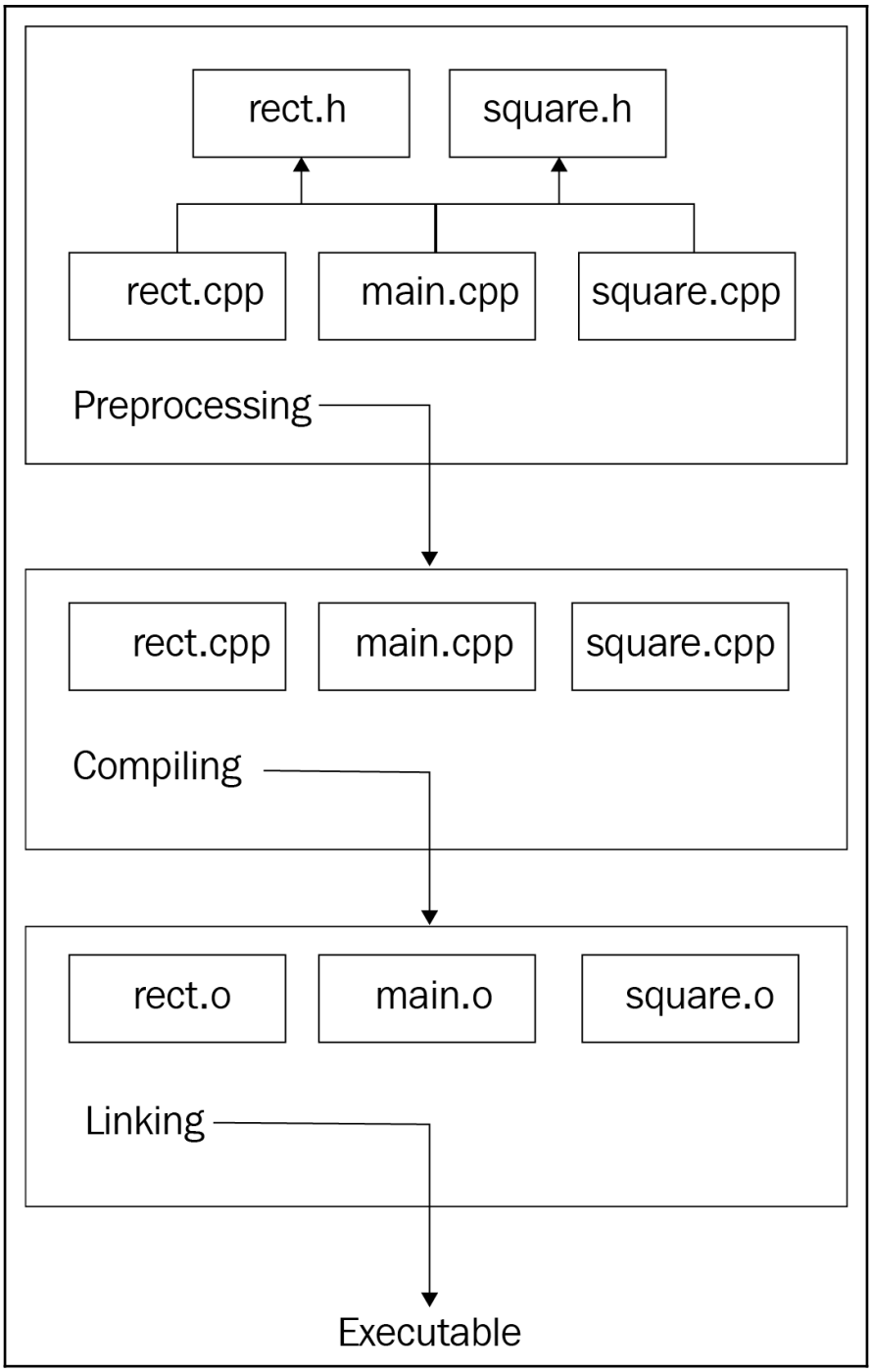
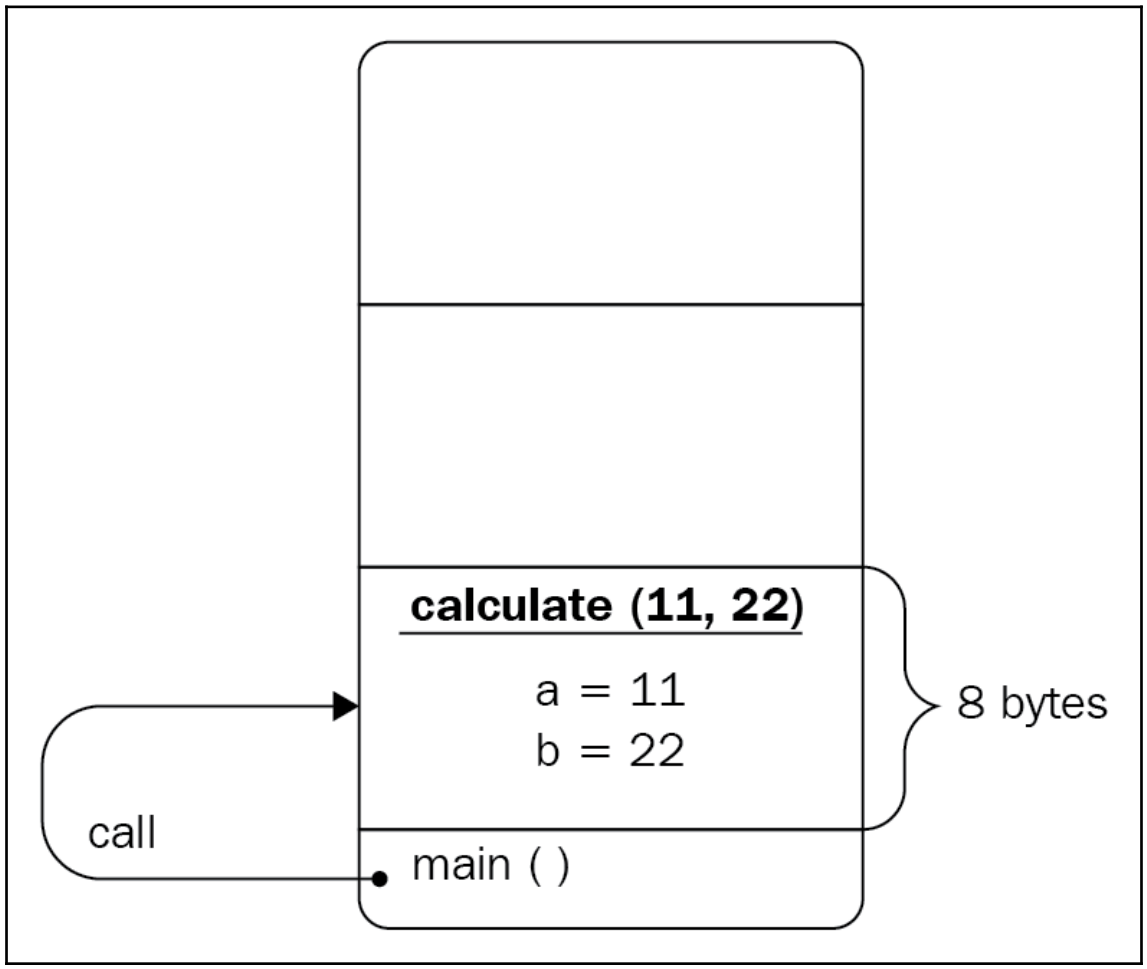
Chapter 1: Introduction to Building C++ Applications

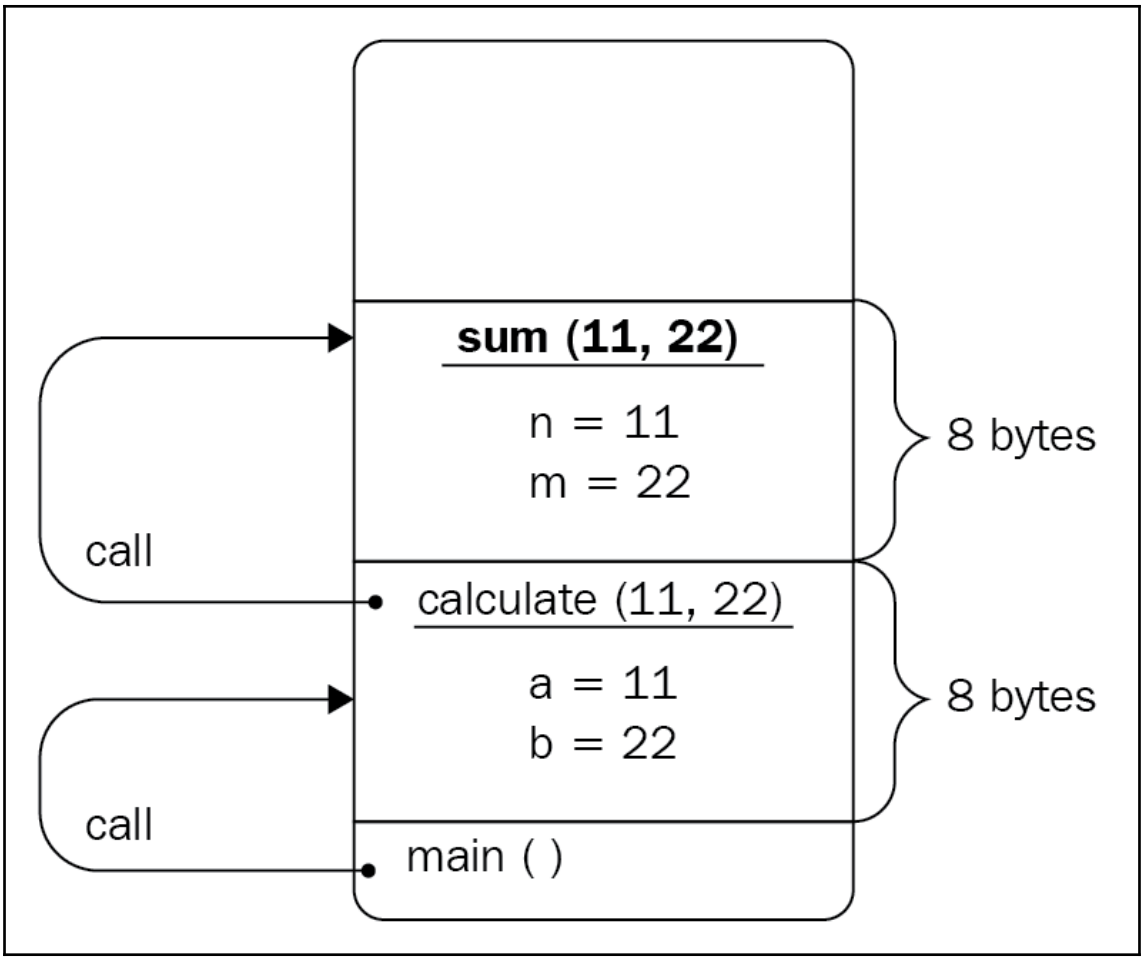


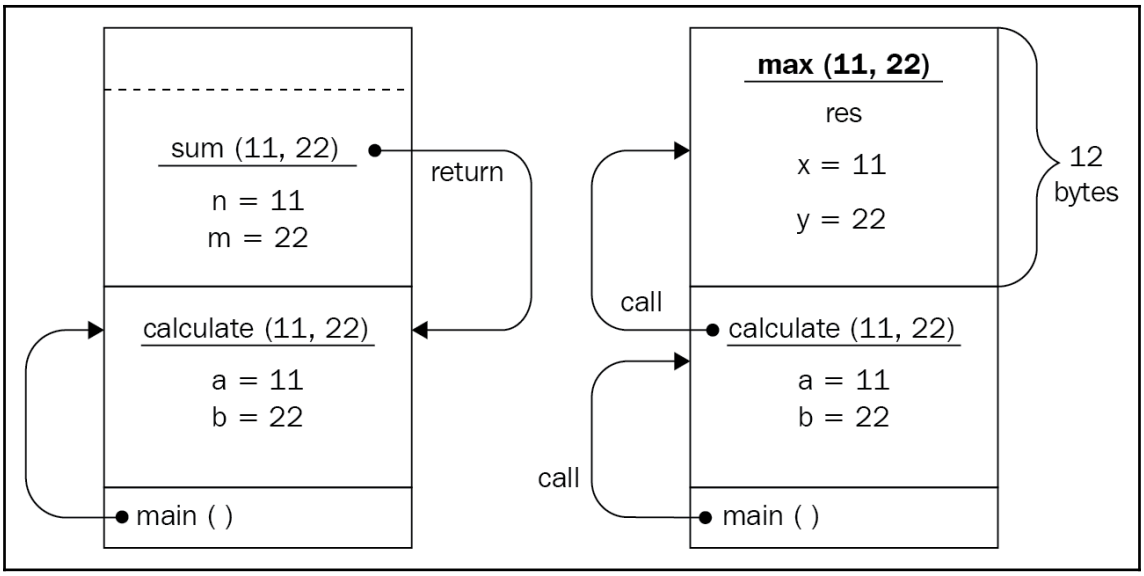

Tokenization
Syntax analysis
Semantic analysis
Intermediate code generation
Optimization
Machine code generation

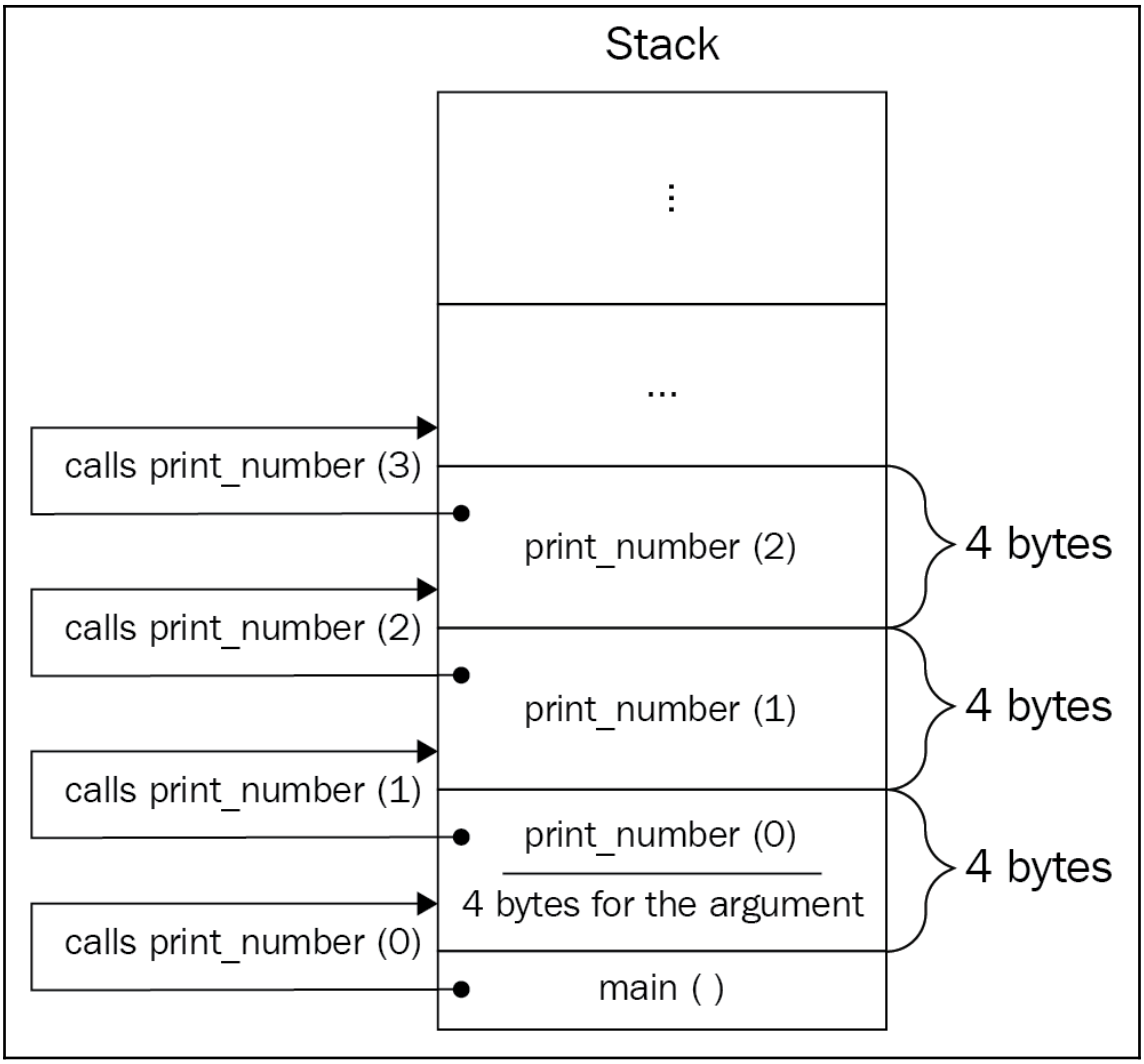

rect.h
square.h
rect.cpp
main.cpp
square.cpp
Preprocessing
rect.cpp
main.cpp
square.cpp
Compiling
rect.o
main.o
square.o
Linking
Executable

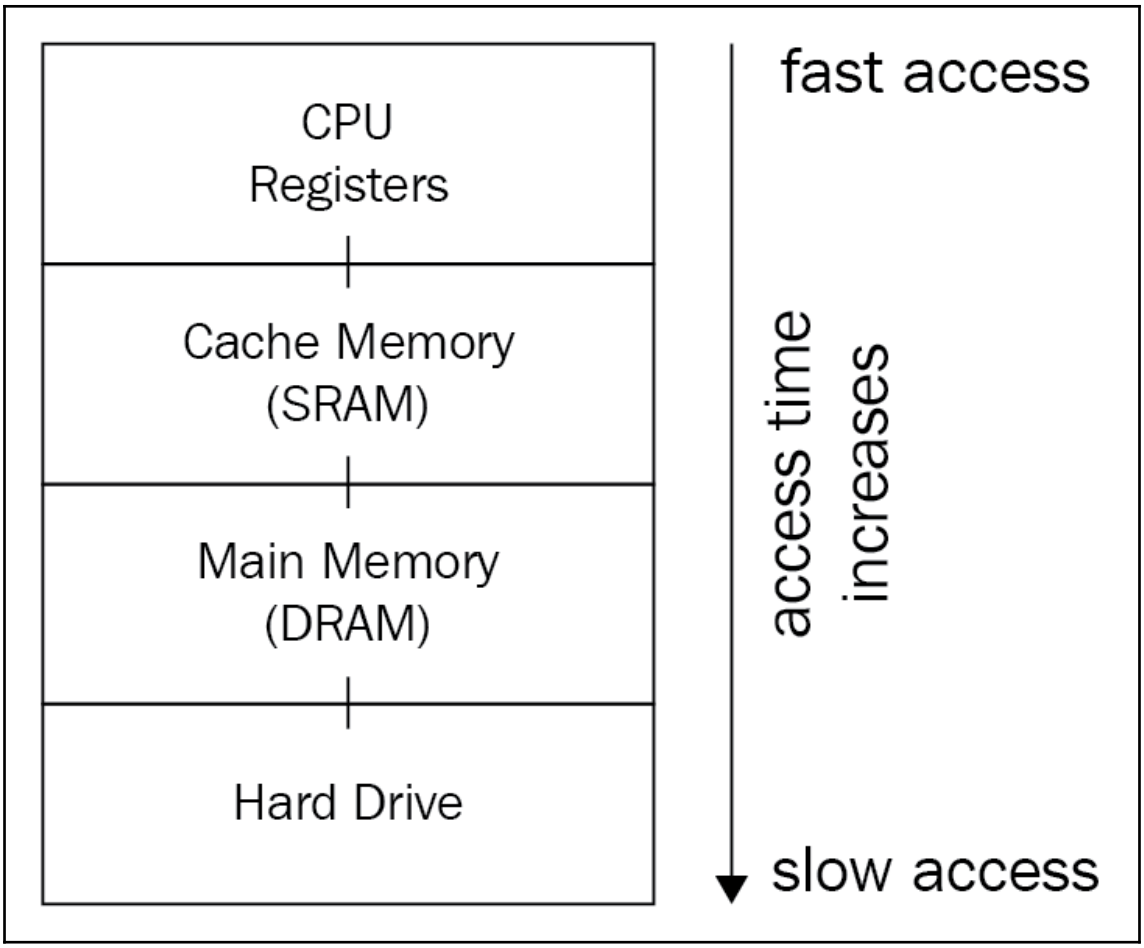
Chapter 2: Low-Level Programming with C++

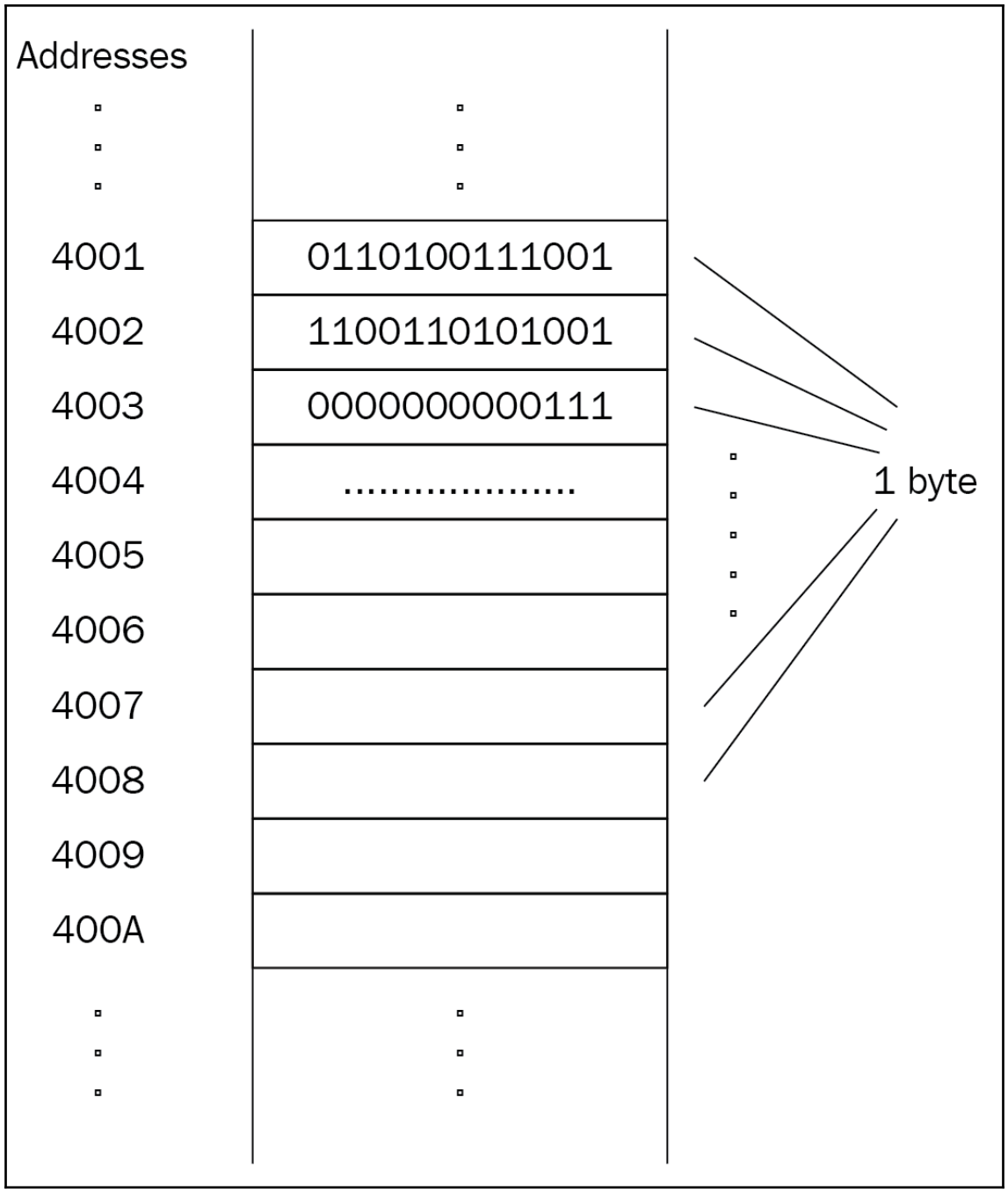


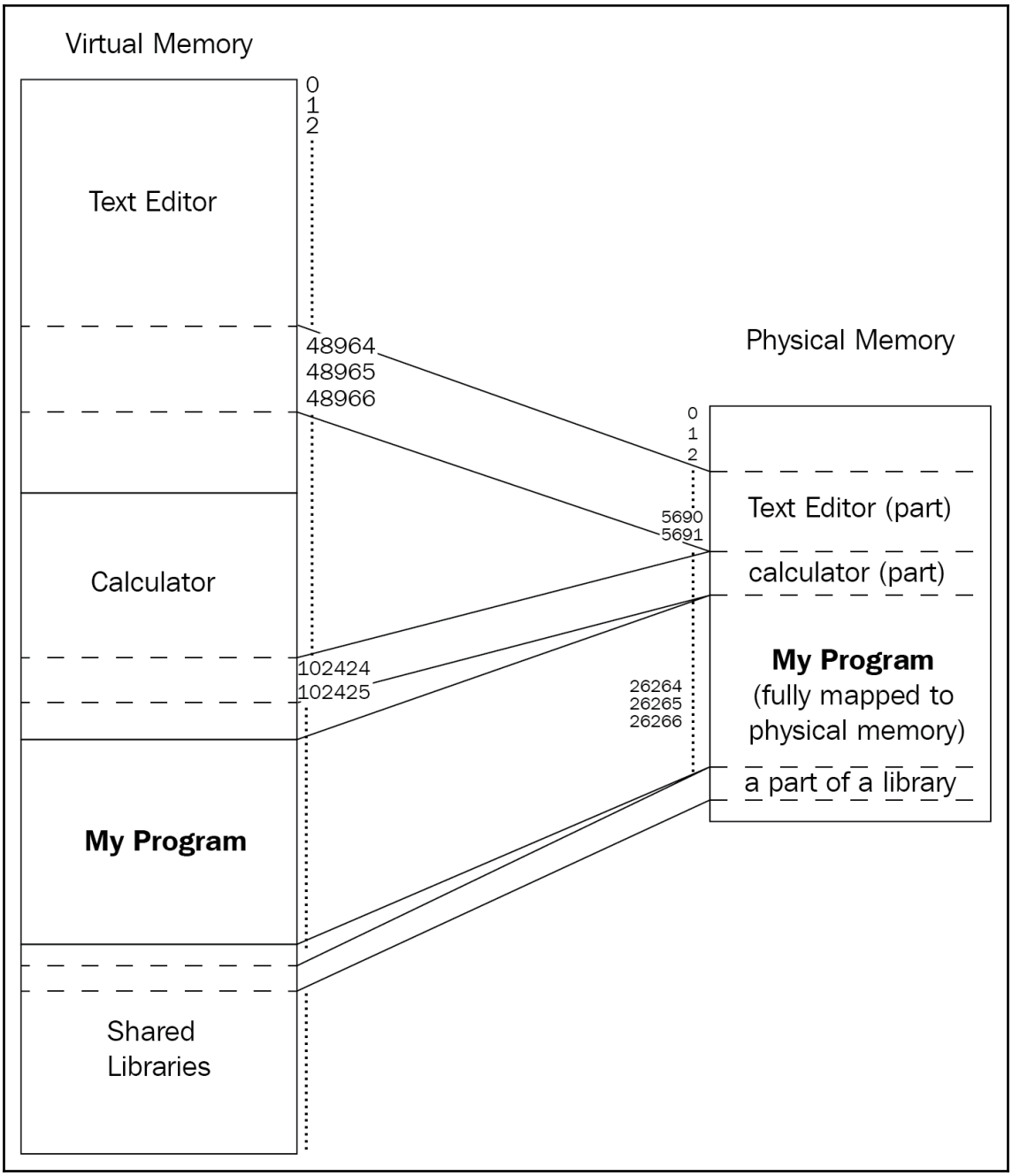

sum (11, 22)
n = 11
m = 22
8 bytes
call
calculate (11, 22)
a = 11
b = 22
8 bytes
call
main ( )


sum (11, 22)
n = 11
m = 22
return
calculate (11, 22)
a = 11
b = 22
main ( )
max (11, 22)
res
x = 11
y = 22
12
bytes
call
calculate (11, 22)
a = 11
b = 22
call
main ( )

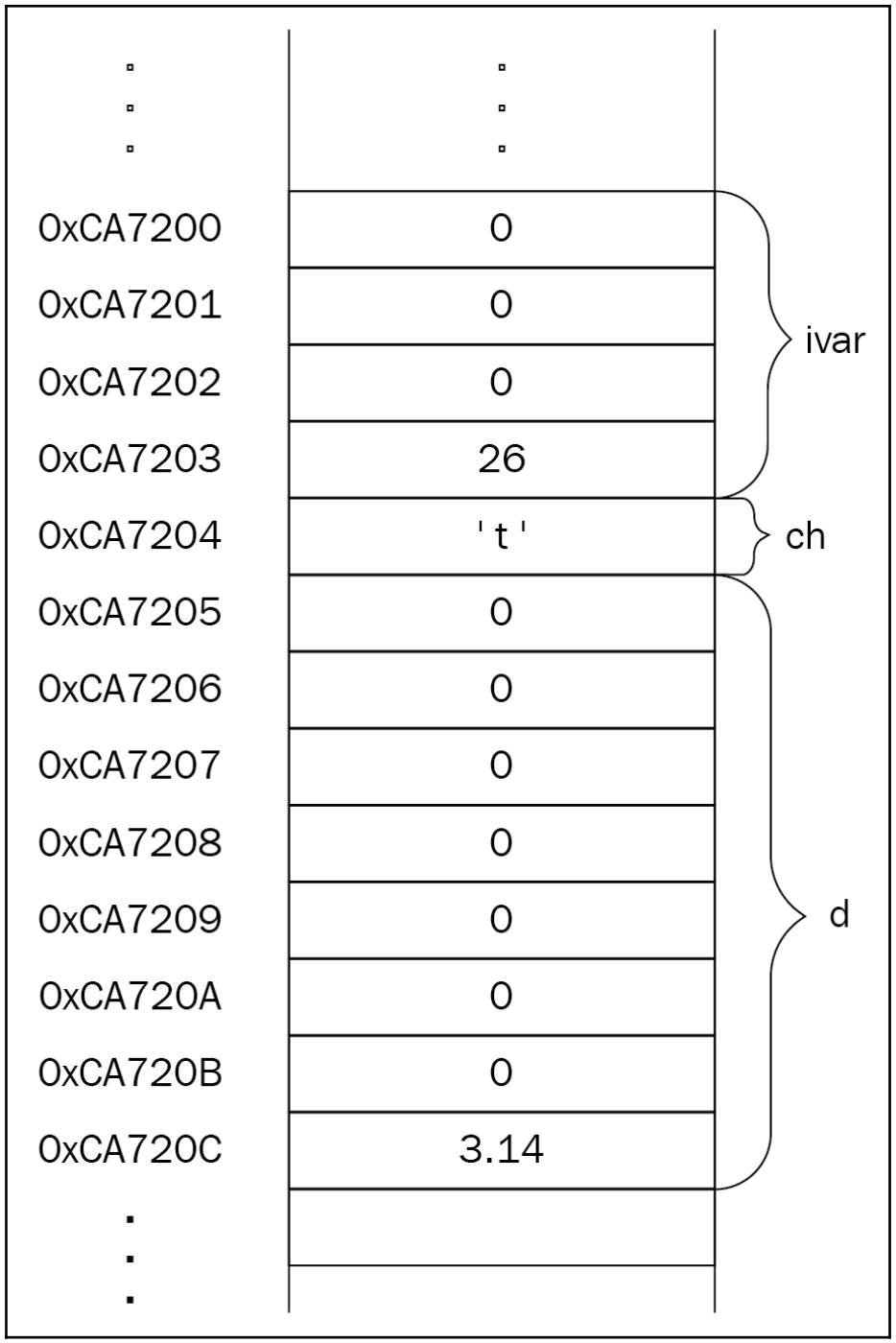

Stack
...
calls print_number (3)
print_number (2)
4 bytes
calls print_number (2)
print_number (1)
4 bytes
calls print_number (1)
print_number (0)
4 bytes for the argument
4 bytes
calls print_number (0)
main ( )

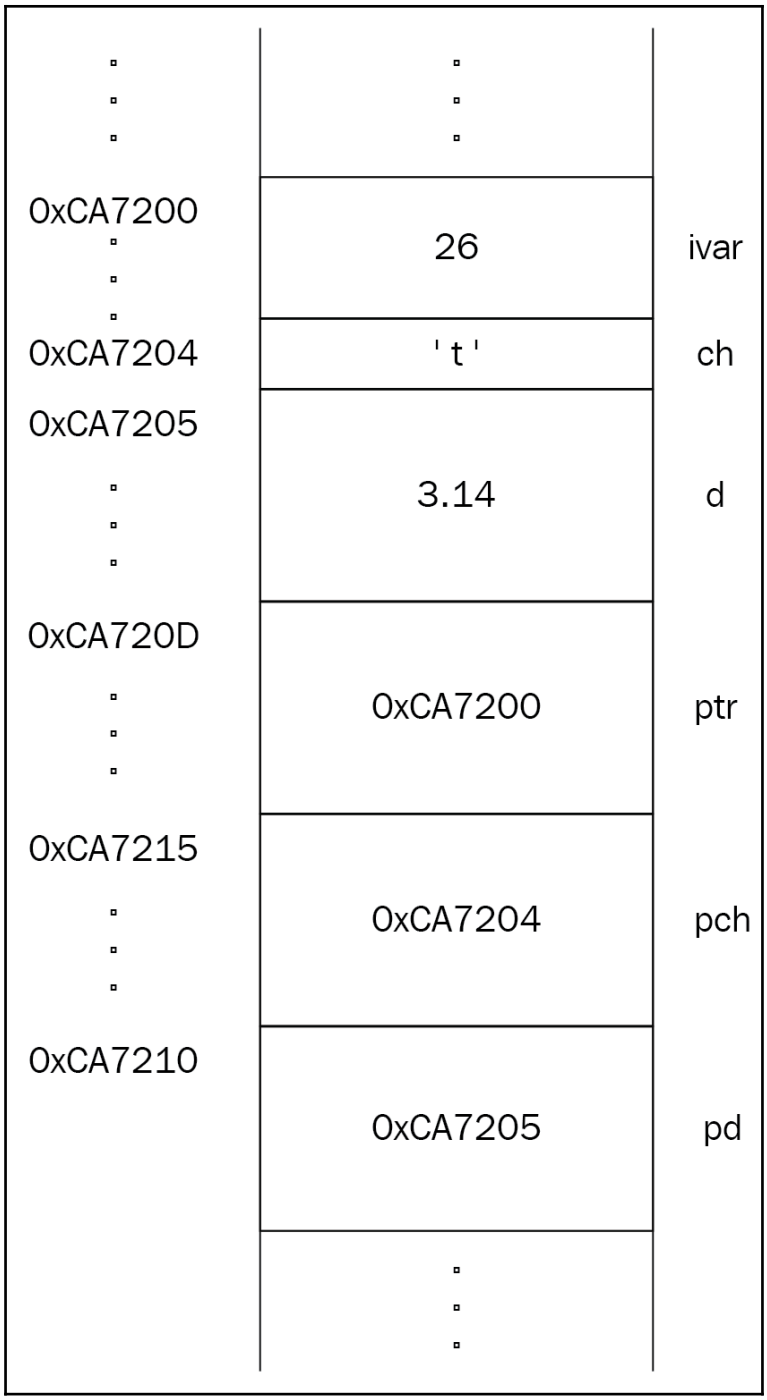

CPU
Registers
Cache Memory
(SRAM)
Main Memory
(DRAM)
Hard Drive
fast access
access time
increases
slow access

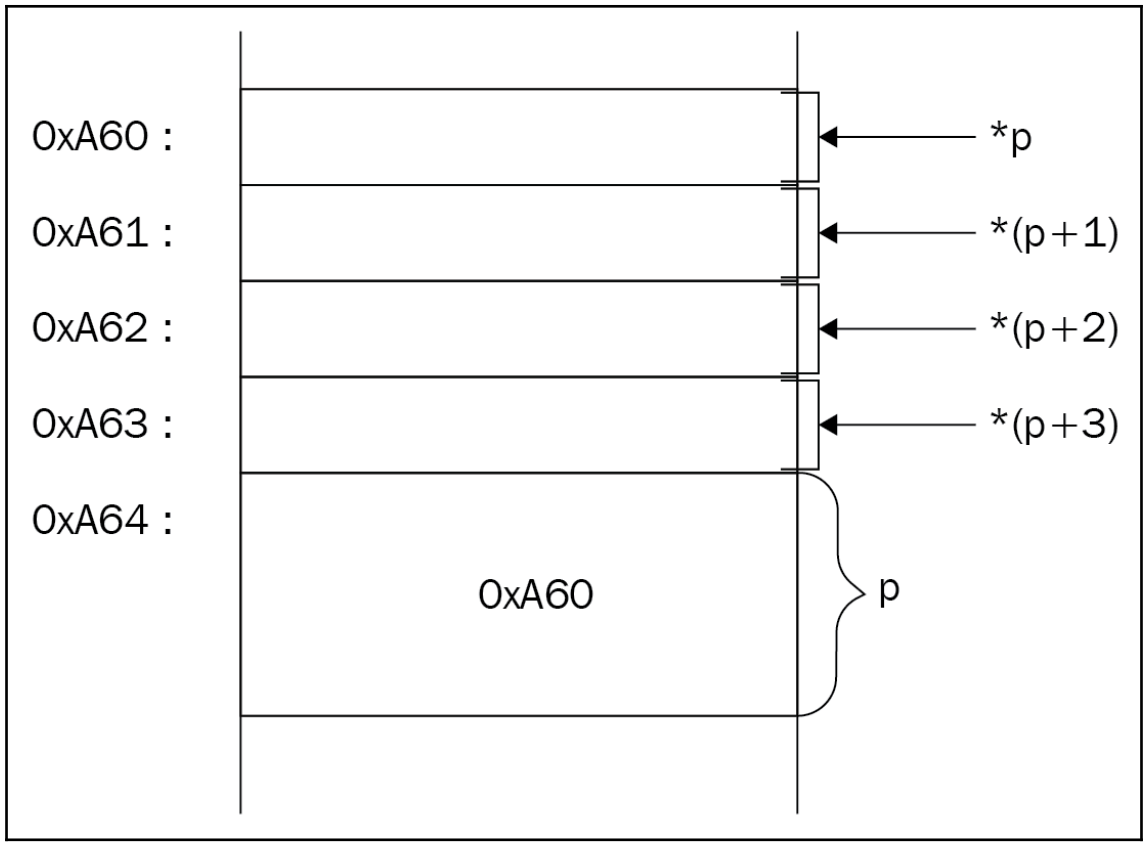

Addresses
4001
0110100111001
4002
1100110101001
4003
0000000000111
4004
.....
4005
4006
4007
4008
4009
400A
1 byte

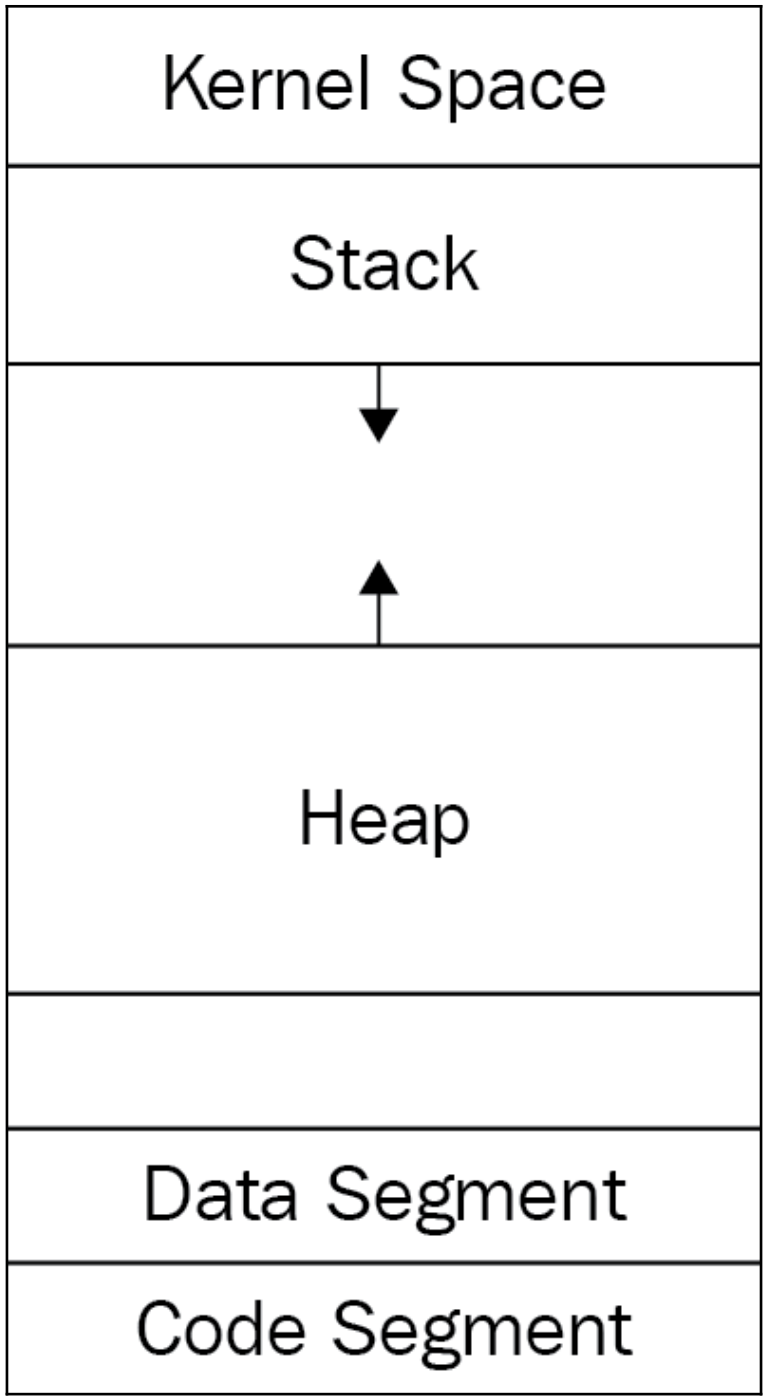

Virtual Memory
0
1
2
Text Editor
48964
48965
48966
Calculator
102424
102425
My Program
Shared
Libraries
Physical Memory
0
1
2
5690
5691
Text Editor (part)
calculator (part)
My Program
(fully mapped to
physical memory)
26264
26265
26266
a part of a library

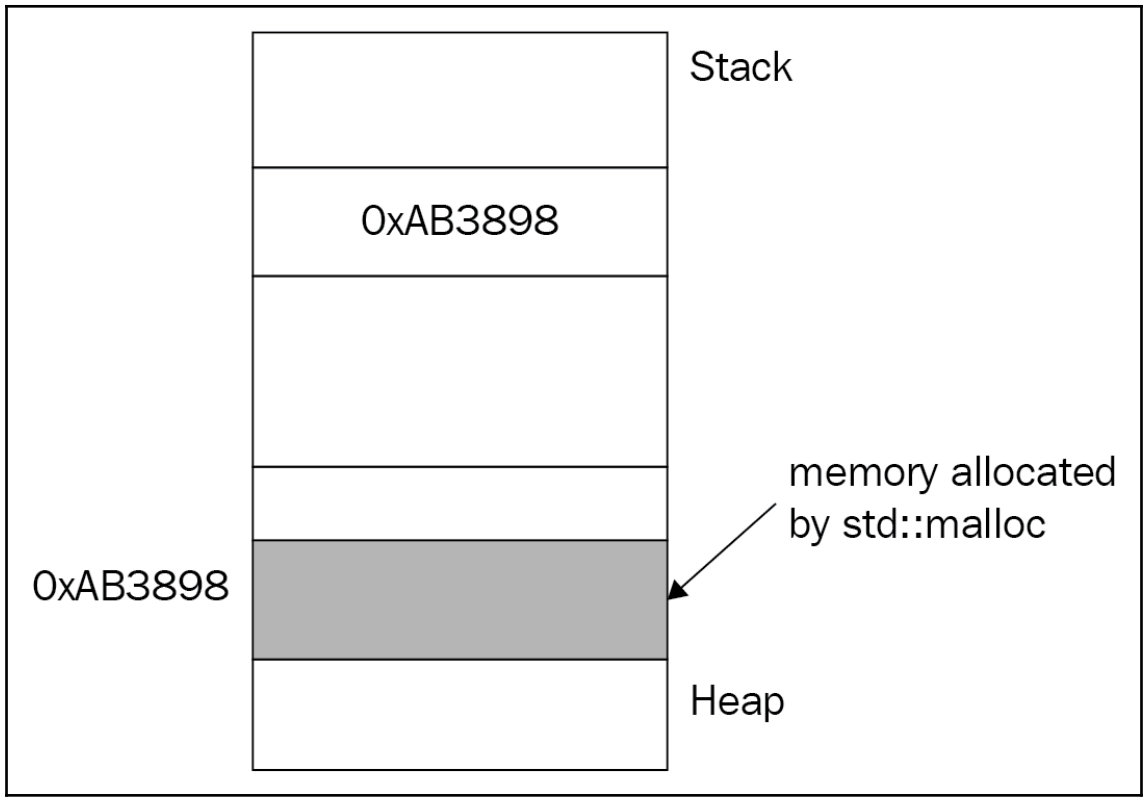
number of bits	number of values	values
1 bit	2	0, 1
2 bit	4	00, 01, 10, 11
3 bit	8	000, 001, 010, 100, ...
4 bit	16	0000, 0001, 0010, ...
.....	.....	.....

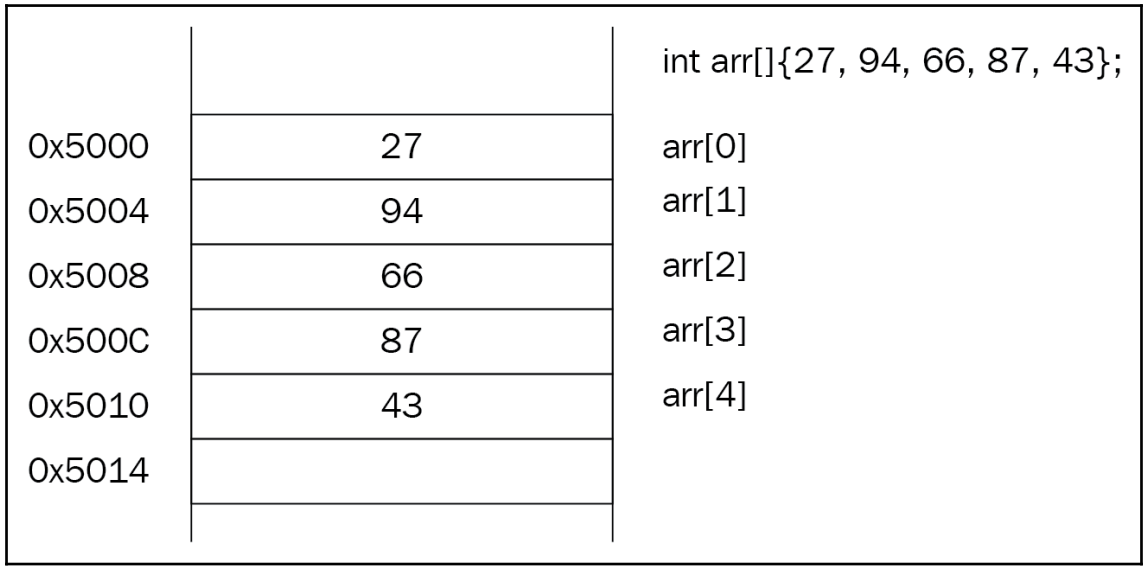

0xCA7200
0xCA7201
0xCA7202
0xCA7203
0xCA7204
0xCA7205
0xCA7206
0xCA7207
0xCA7208
0xCA7209
0xCA720A
0xCA720B
0xCA720C
0
0
0
26
' t '
0
0
0
0
0
0
0
3.14
ivar
ch
d


0xCA7200
26
ivar
0xCA7204
' t '
ch
0xCA7205
3.14
d
0xCA720D
0xCA7200
ptr
0xCA7215
0xCA7204
pch
0xCA7210
0xCA7205
pd

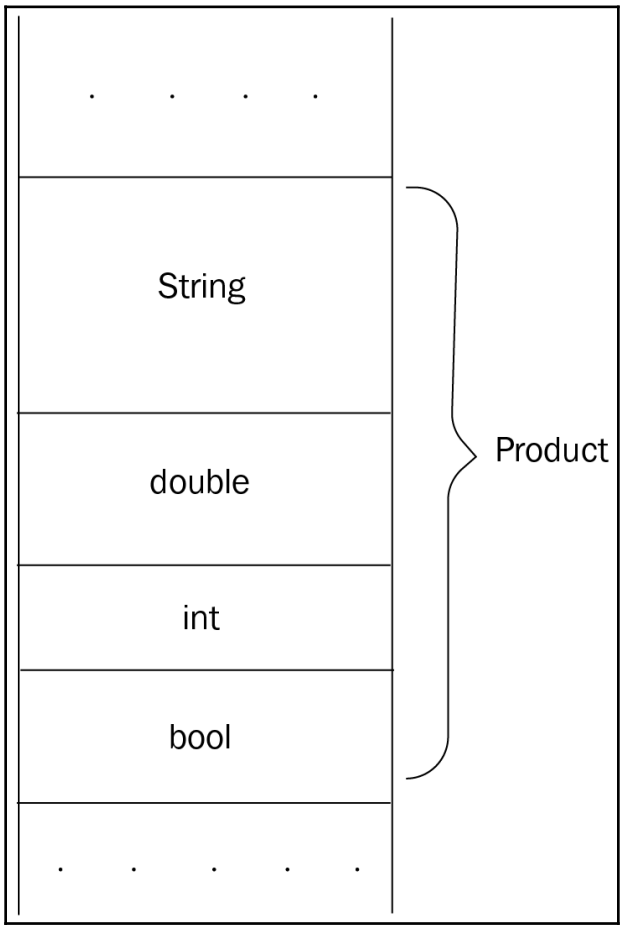

0xA60 :
0xA61 :
0xA62 :
0xA63 :
0xA64 :
0xA60
*p
*(p+1)
*(p+2)
*(p+3)
p

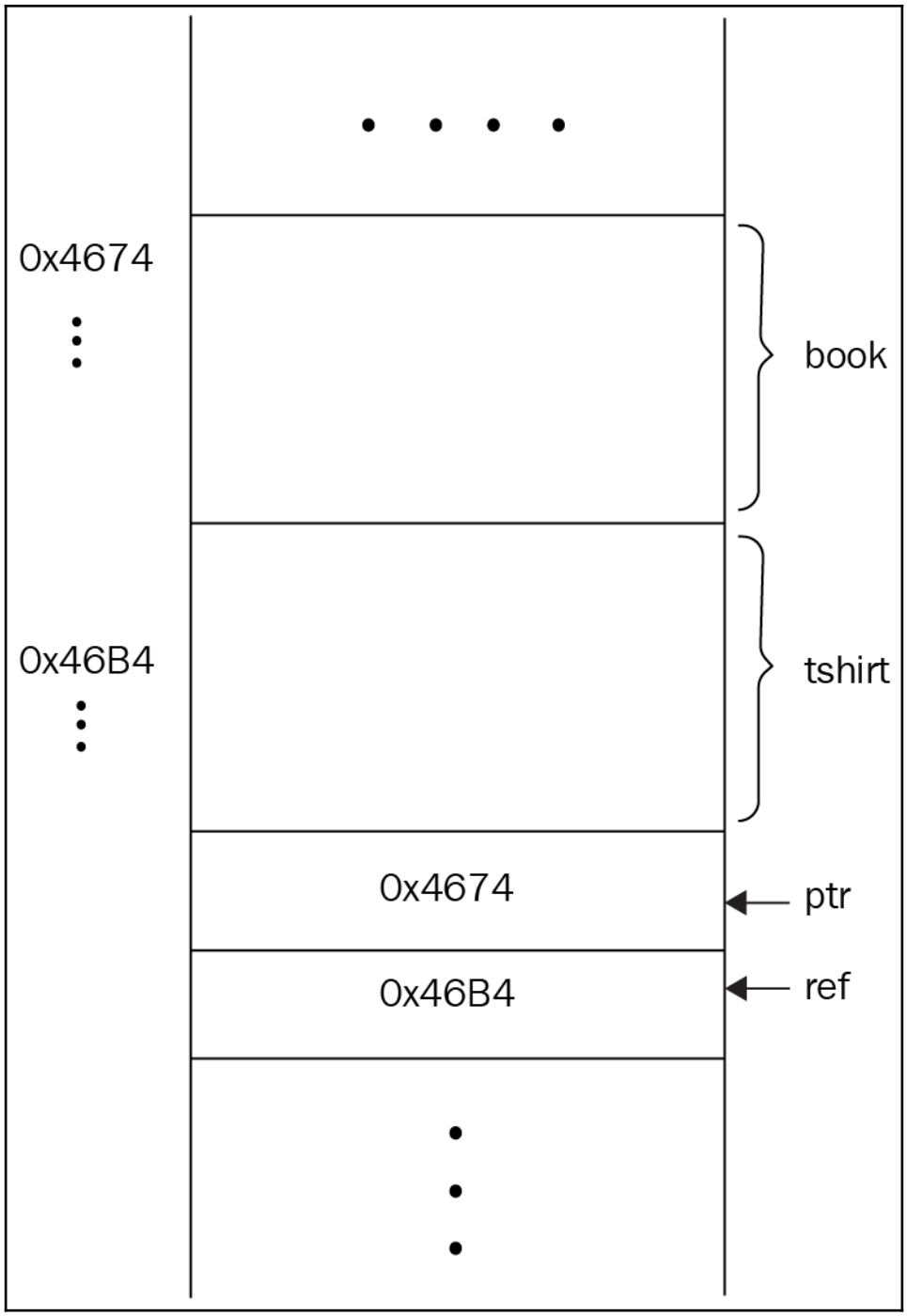

Kernel Space
Stack
Heap
Data Segment
Code Segment

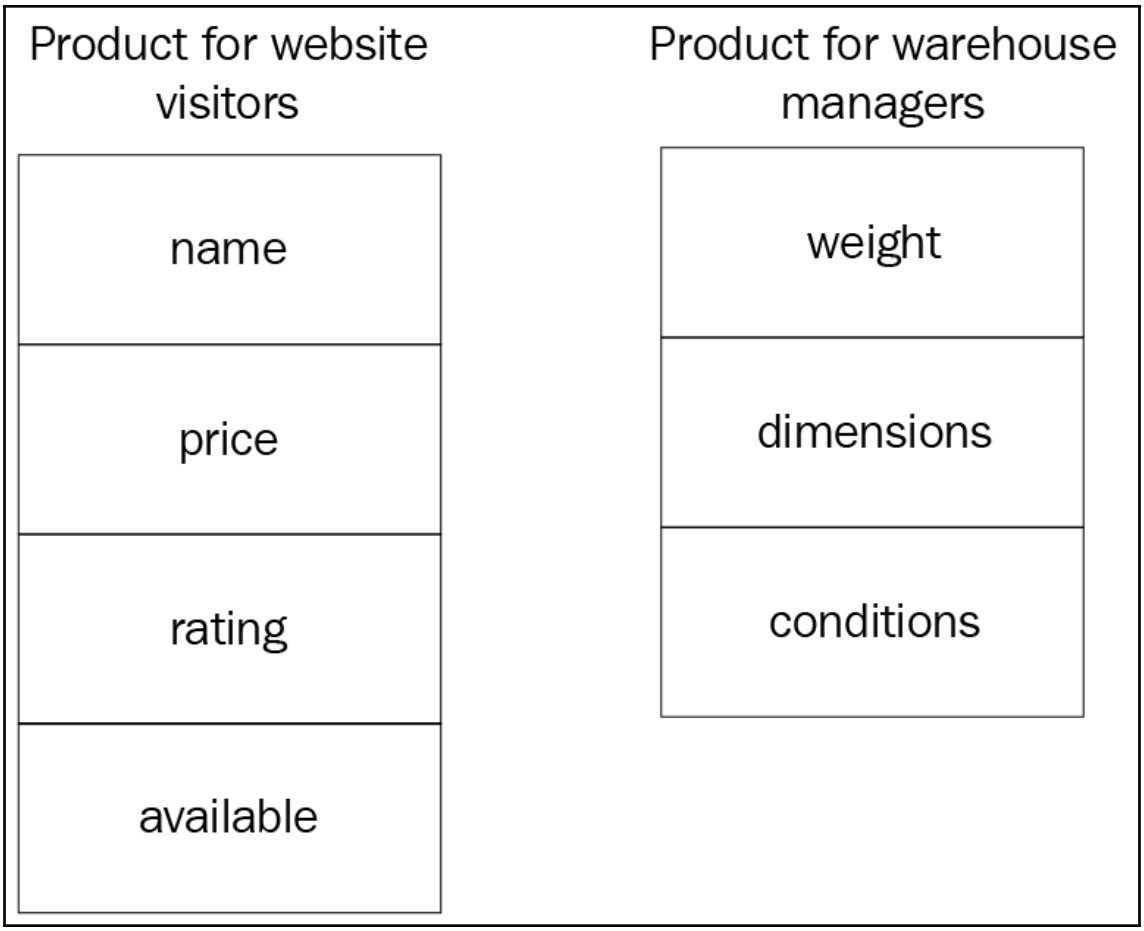

Stack
0xAB3898
memory allocated
by std::malloc
0xAB3898
Heap

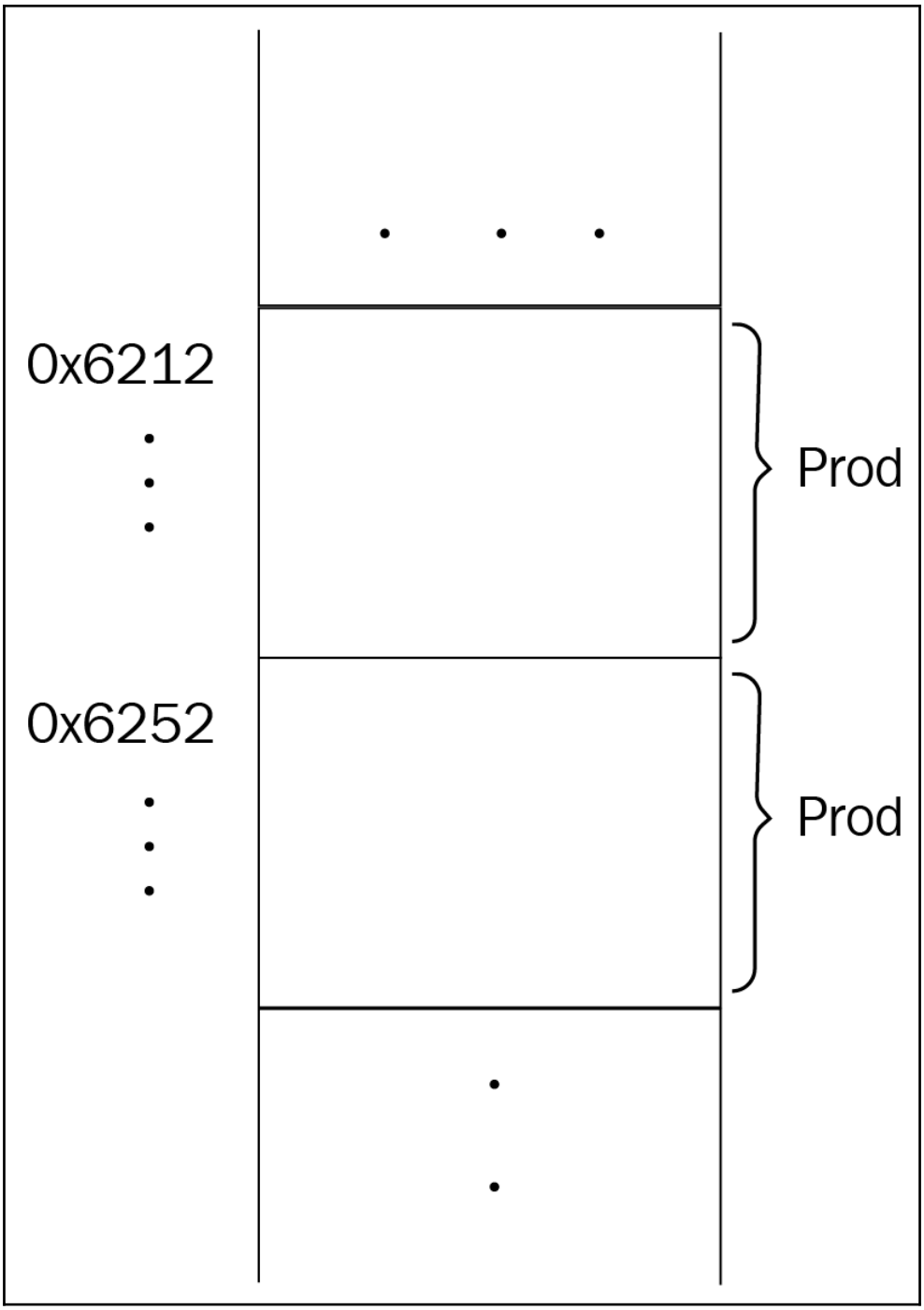

int arr[]{27, 94, 66, 87, 43};
0x5000
27
arr[0]
0x5004
94
arr[1]
0x5008
66
arr[2]
0x500C
87
arr[3]
0x5010
43
arr[4]
0x5014

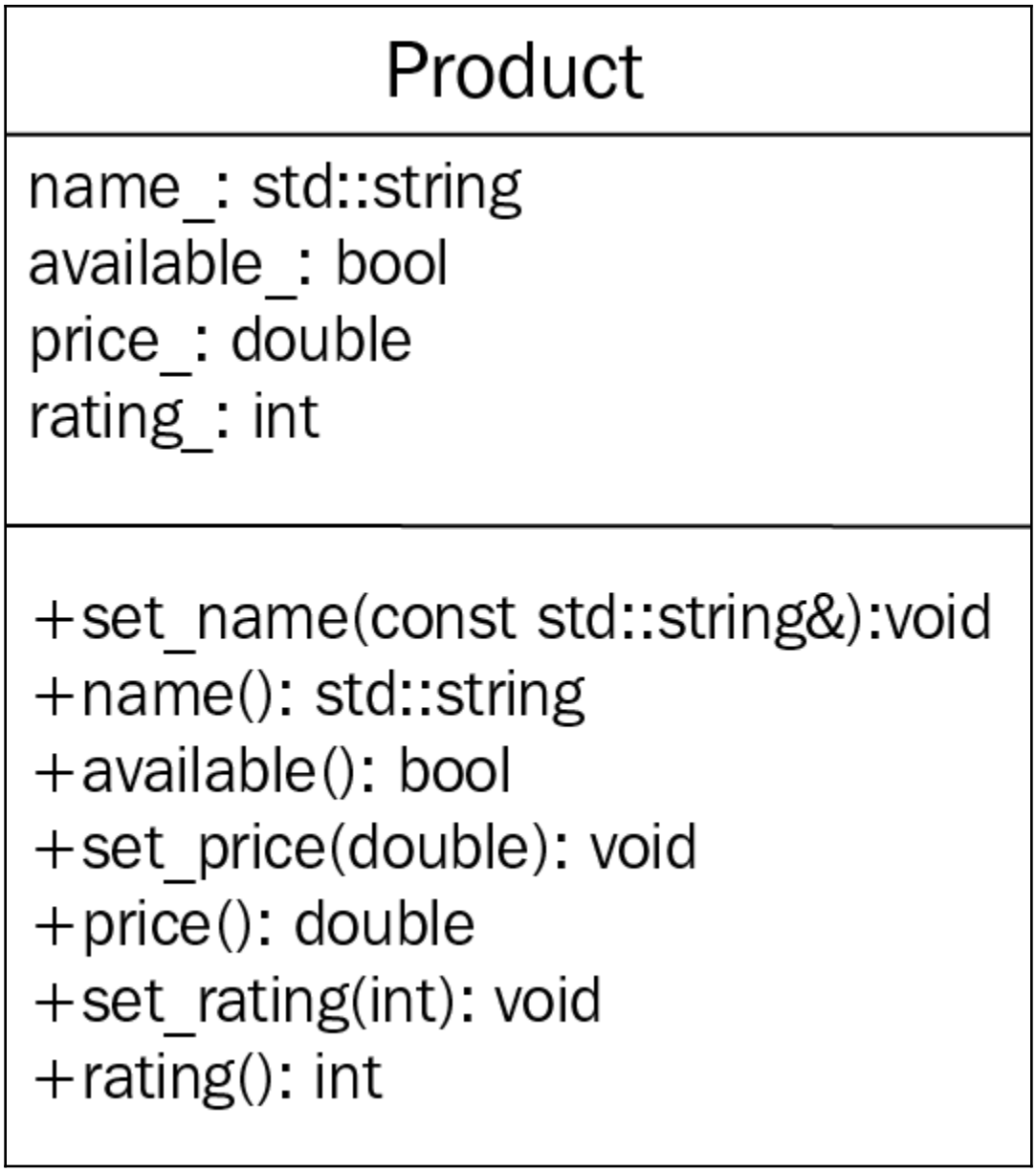
Chapter 3: Details of Object-Oriented Programming



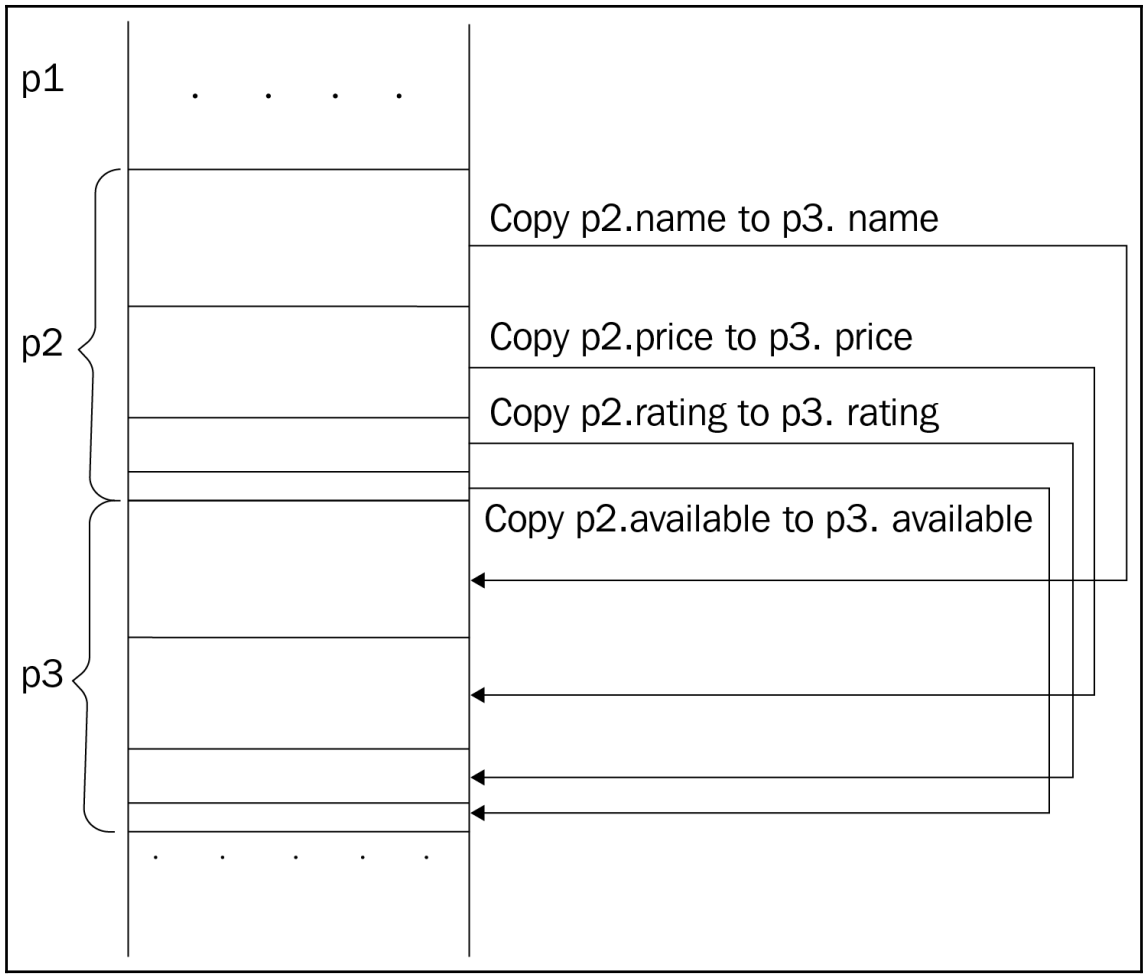

0x4674
book
0x46B4
tshirt
0x4674
ptr
0x46B4
ref

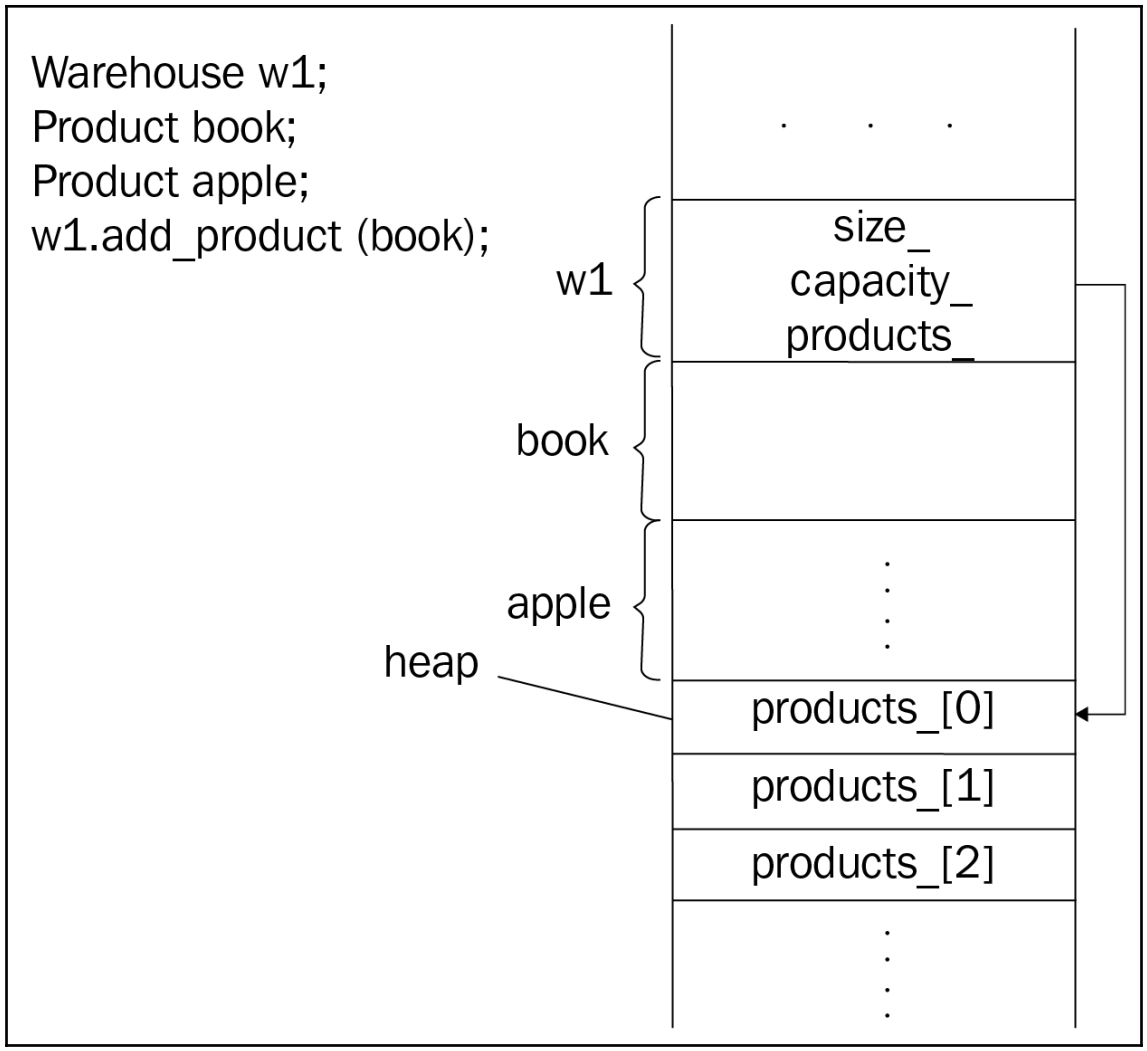

Product for website visitors
name
price
rating
available
Product for warehouse managers
weight
dimensions
conditions

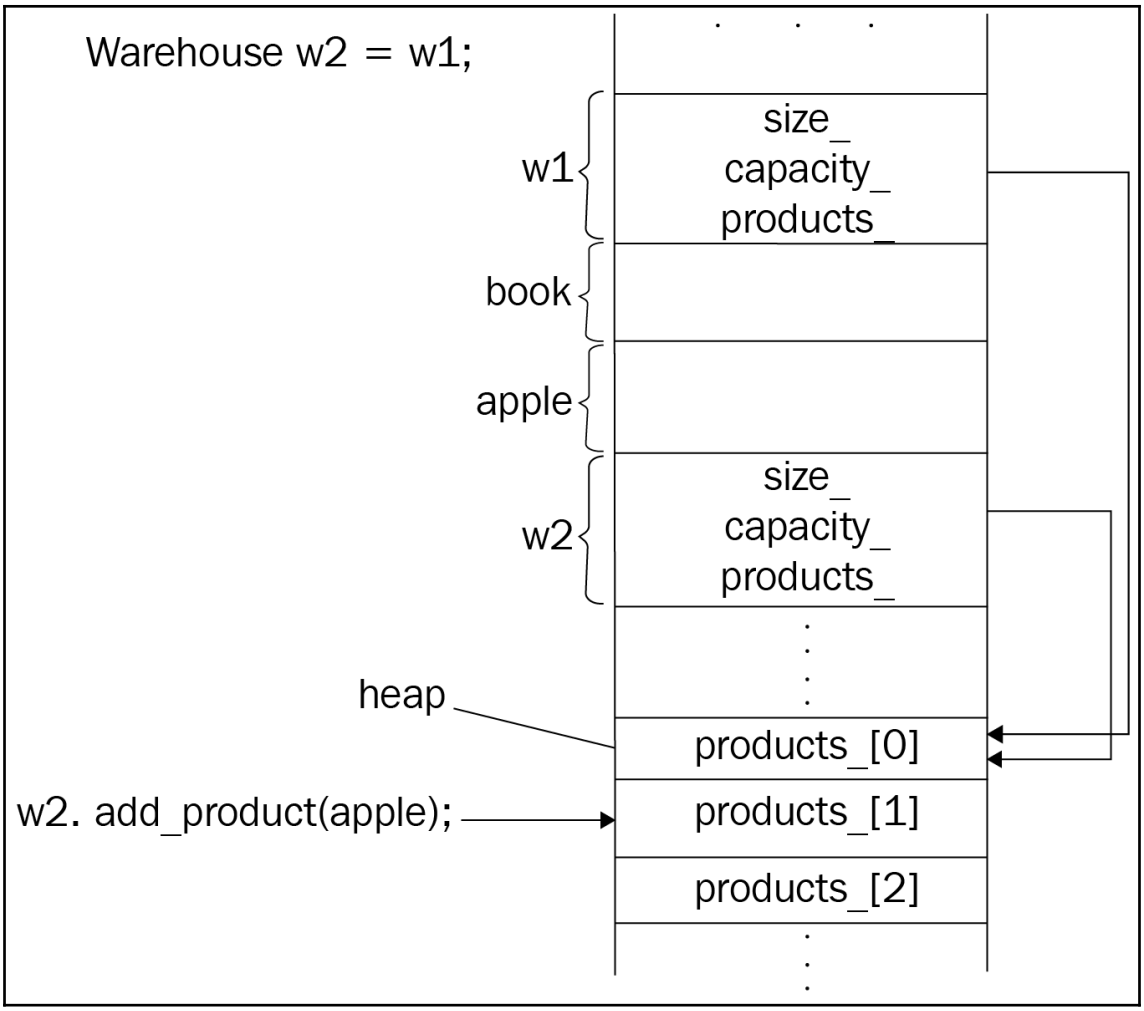

0x6212
Prod
0x6252
Prod

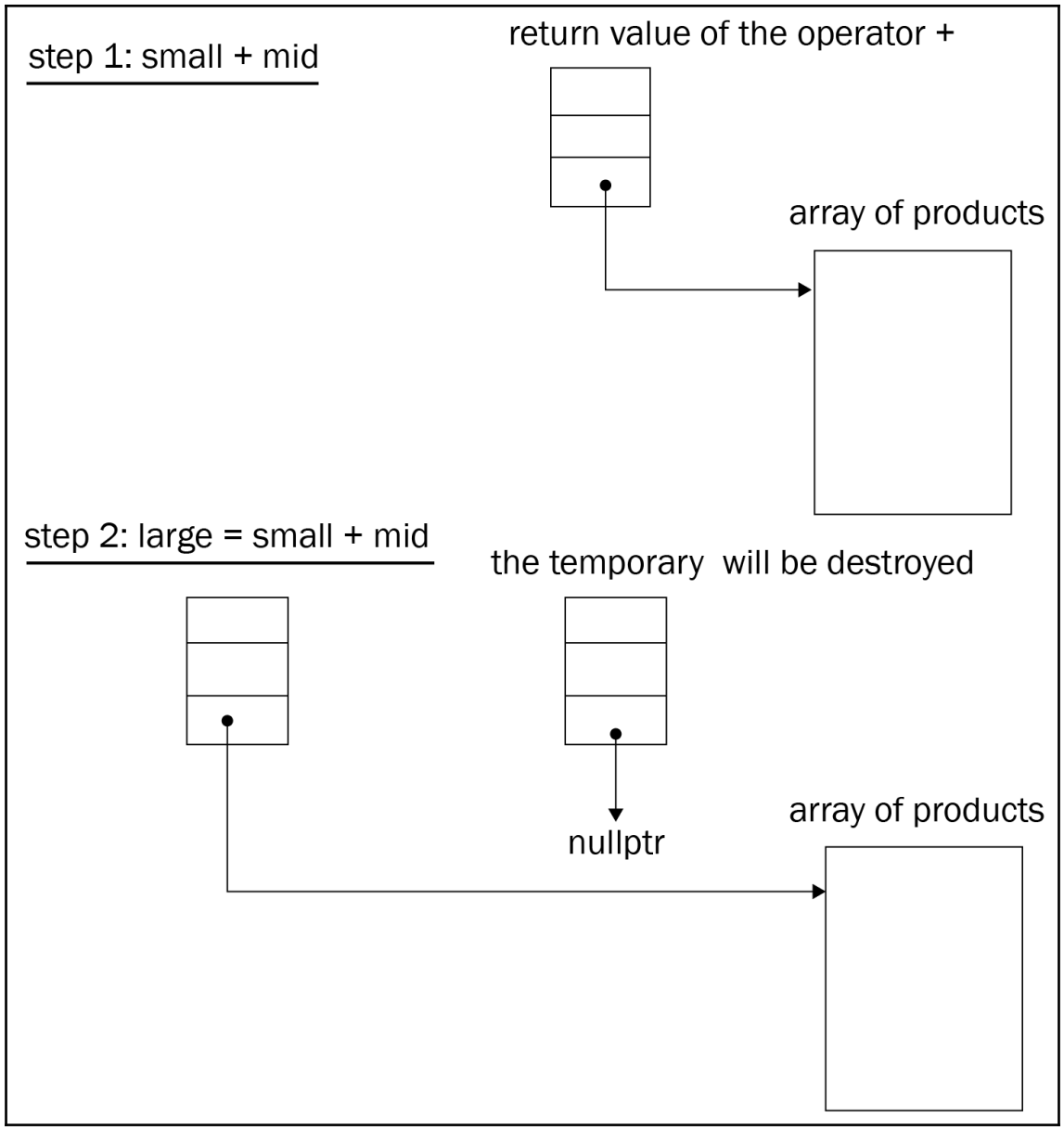

Product
name_: std::string
available_: bool
price_: double
rating_: int
+set_name(const std::string&):void
+name(): std::string
+available(): bool
+set_price(double): void
+price(): double
+set_rating(int): void
+rating(): int

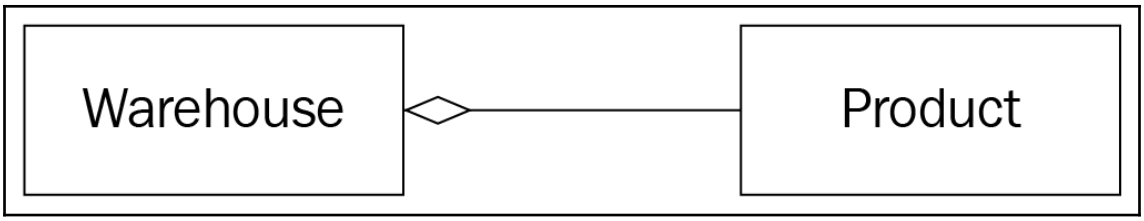
Product

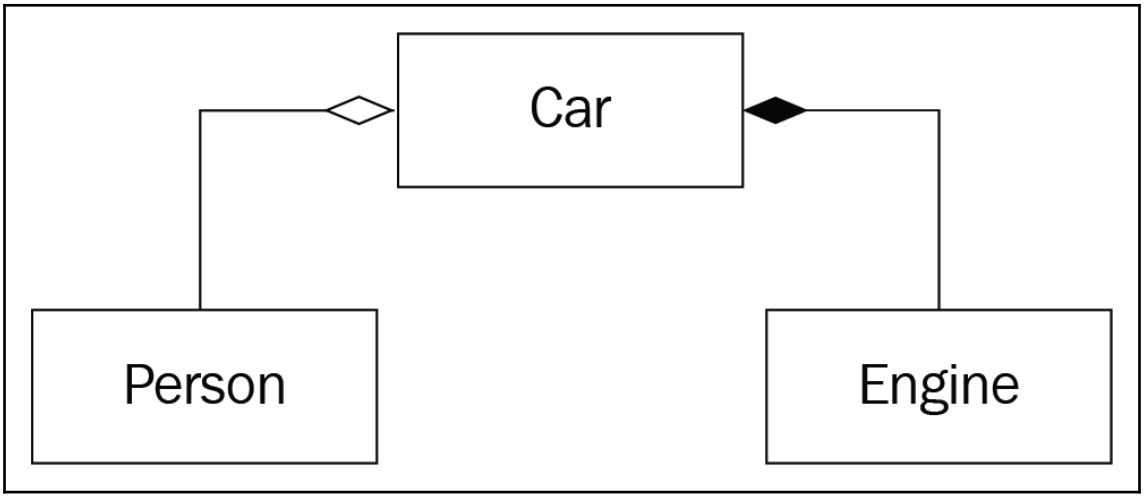

p1
p2
p3
Copy p2.name to p3. name
Copy p2.price to p3. price
Copy p2.rating to p3. rating
Copy p2.available to p3. available

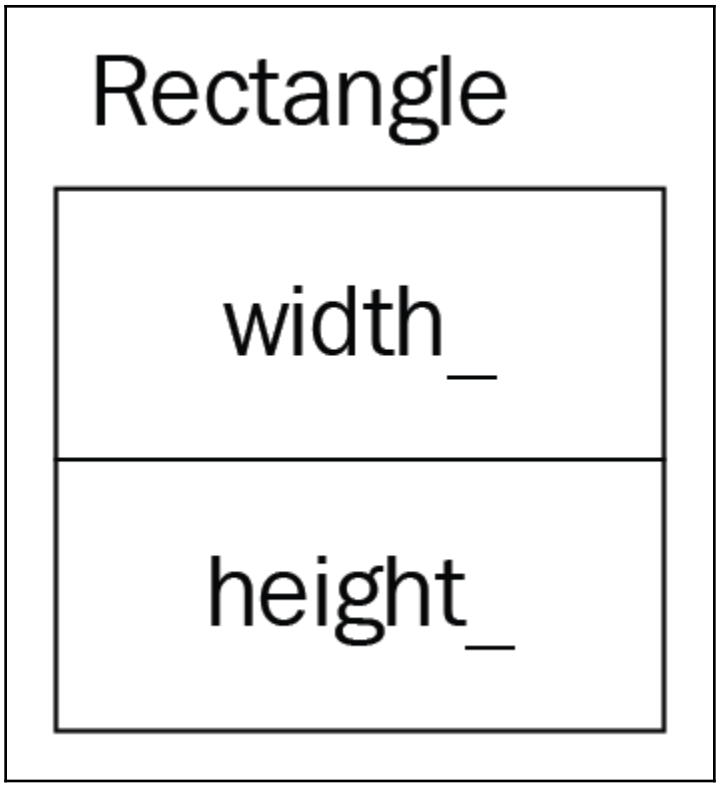

Warehouse w1;
Product book;
Product apple;
w1.add_product (book);
w1
book
apple
heap
size_
capacity_
products_
products_[0]
products_[1]
products_[2]

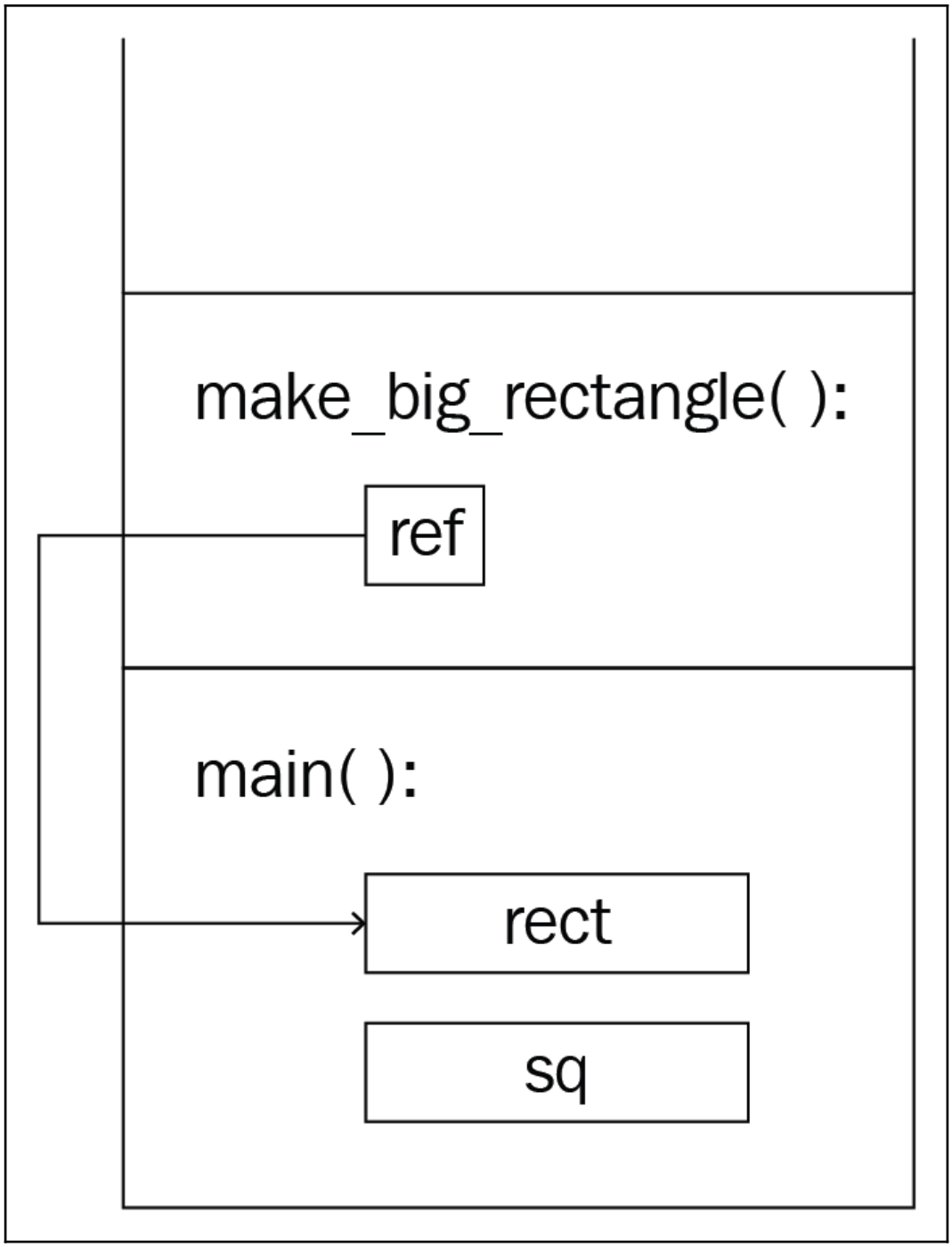

Warehouse w2 = w1;
w1
size_
capacity_
products_
book
apple
w2
size_
capacity_
products_
heap
products_[0]
w2. add_product(apple);
products_[1]
products_[2]

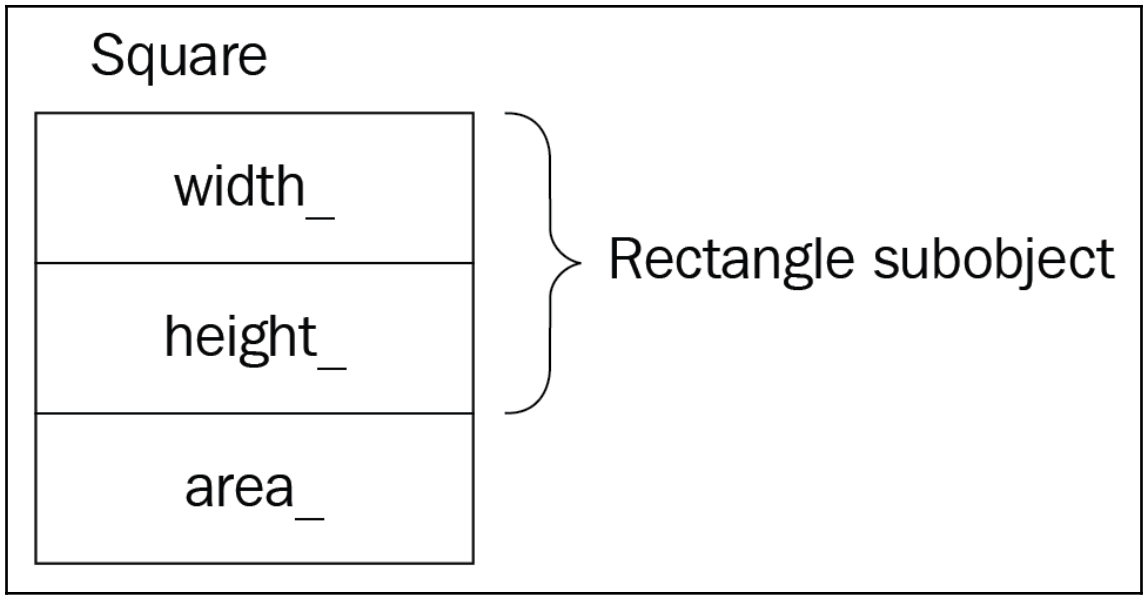

step 1: small + mid
return value of the operator +
array of products
step 2: large = small + mid
the temporary will be destroyed
nullptr
array of products

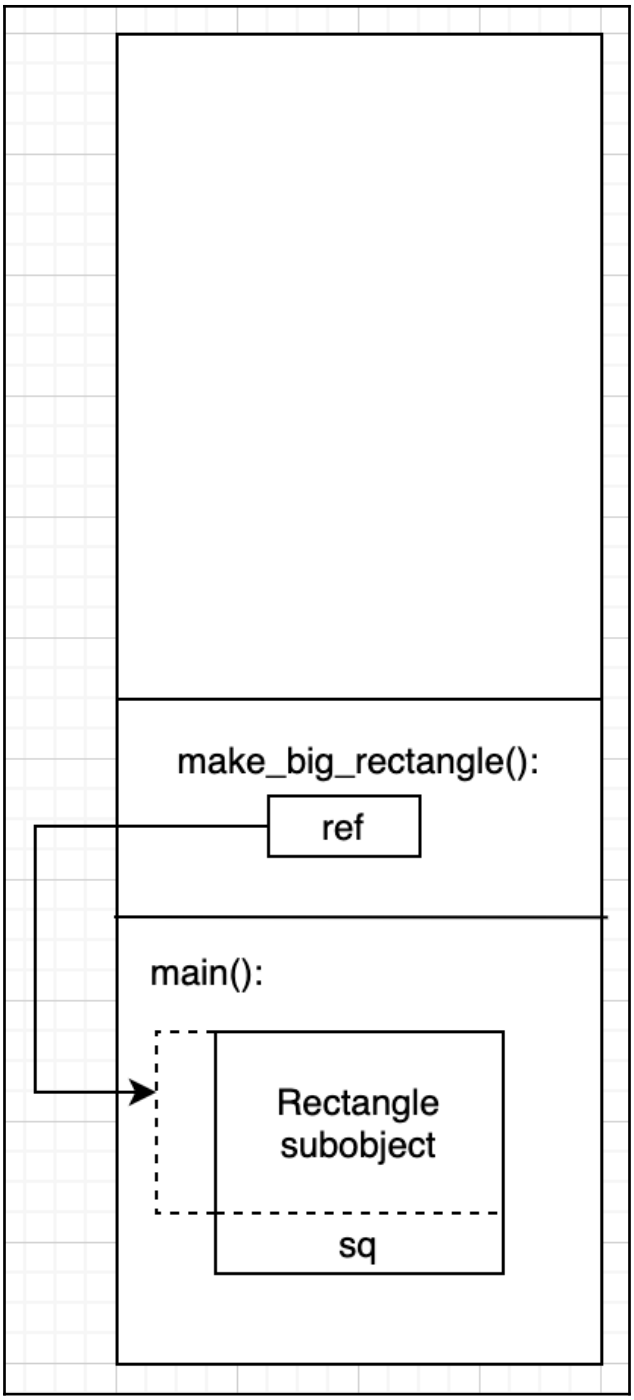

Warehouse
Product


Car
Person
Engine

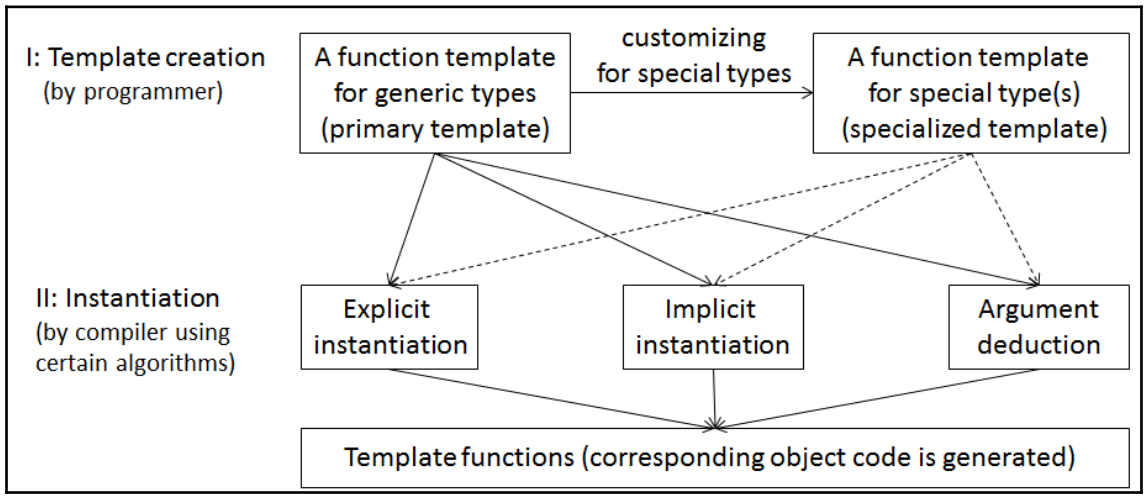

Rectangle
width_
height_

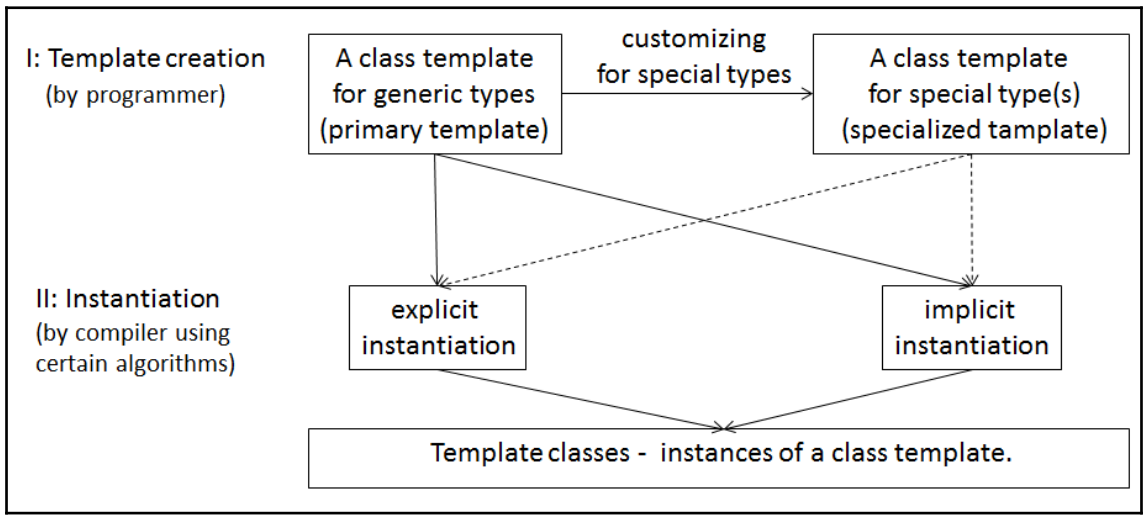

make_big_rectangle( ):
ref
main( ):
rect
sq


Square
width_
height_
area_
Rectangle subobject

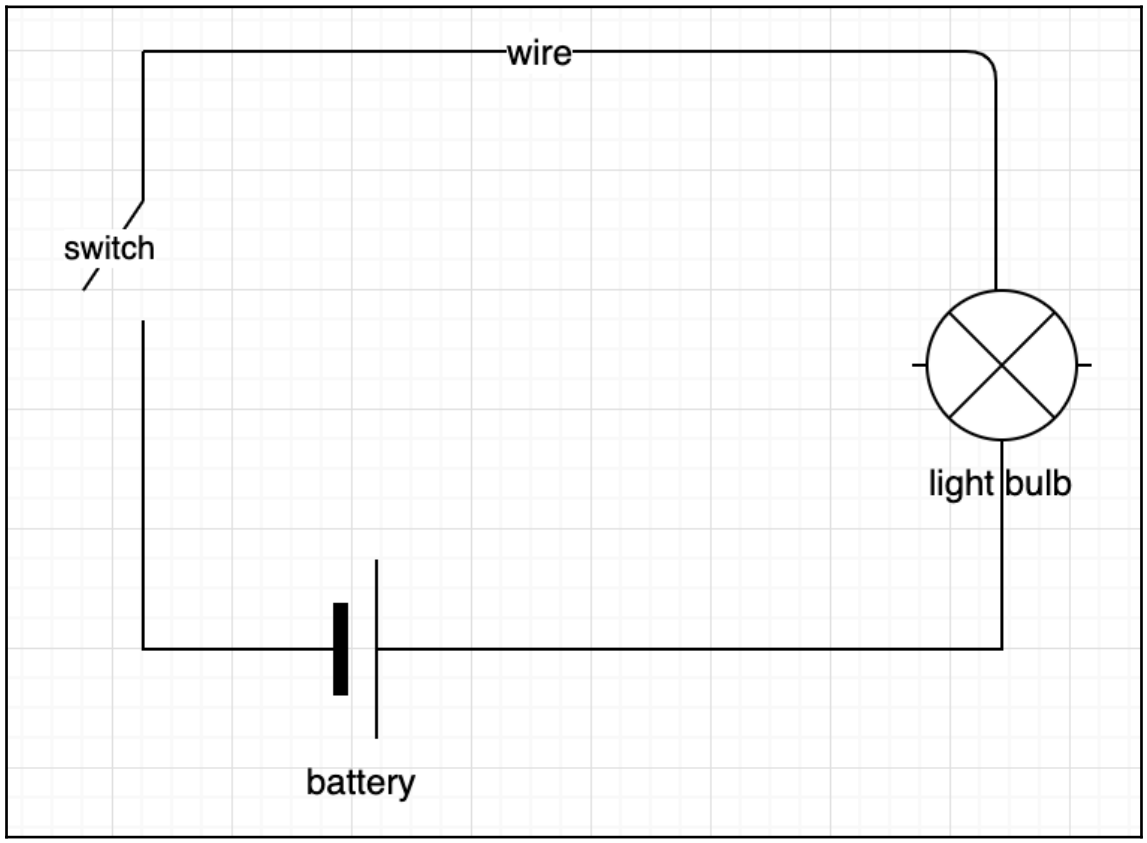

make_big_rectangle():
ref
main():
Rectangle
subobject
sq

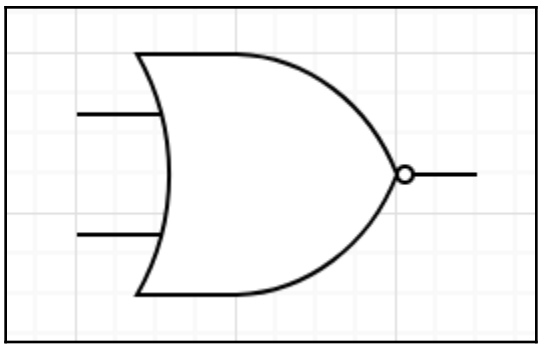
Chapter 4: Understanding and Designing Templates

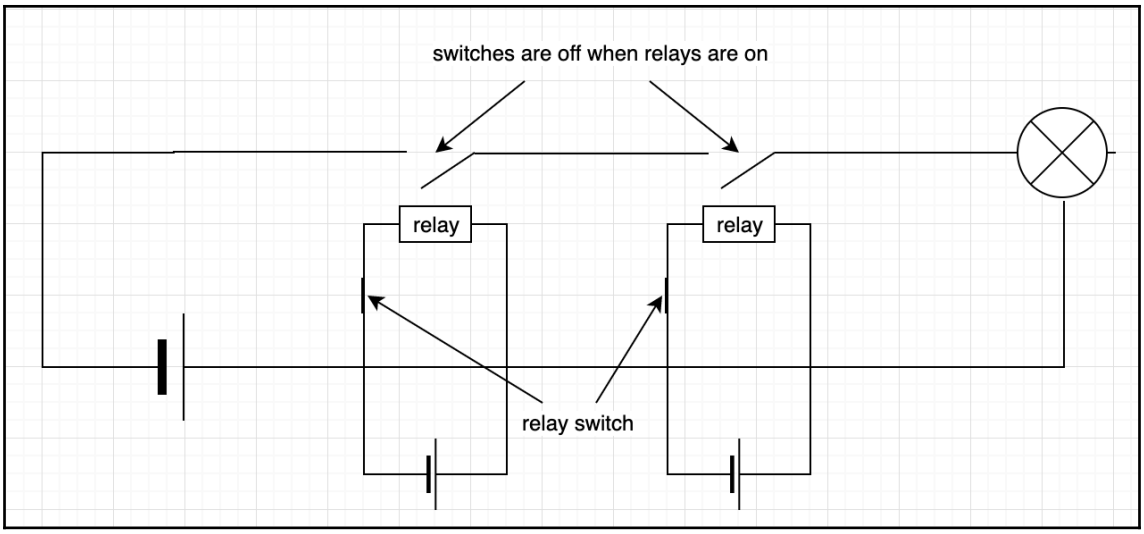


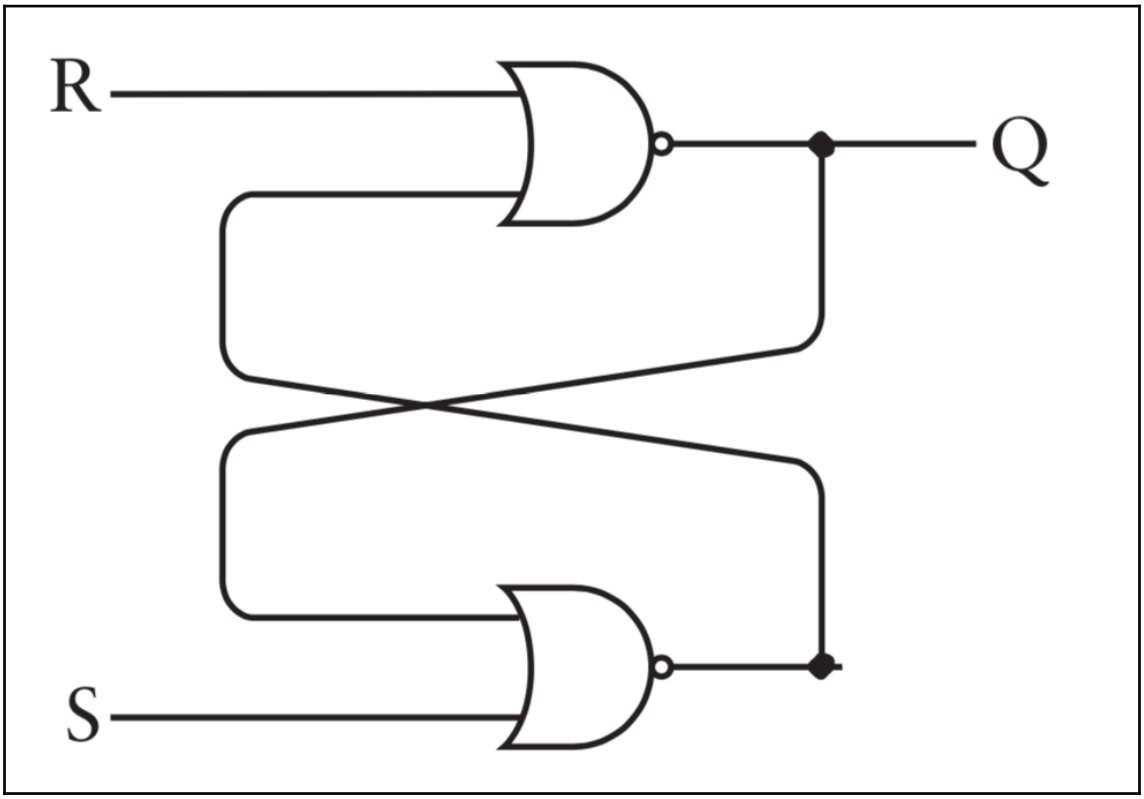


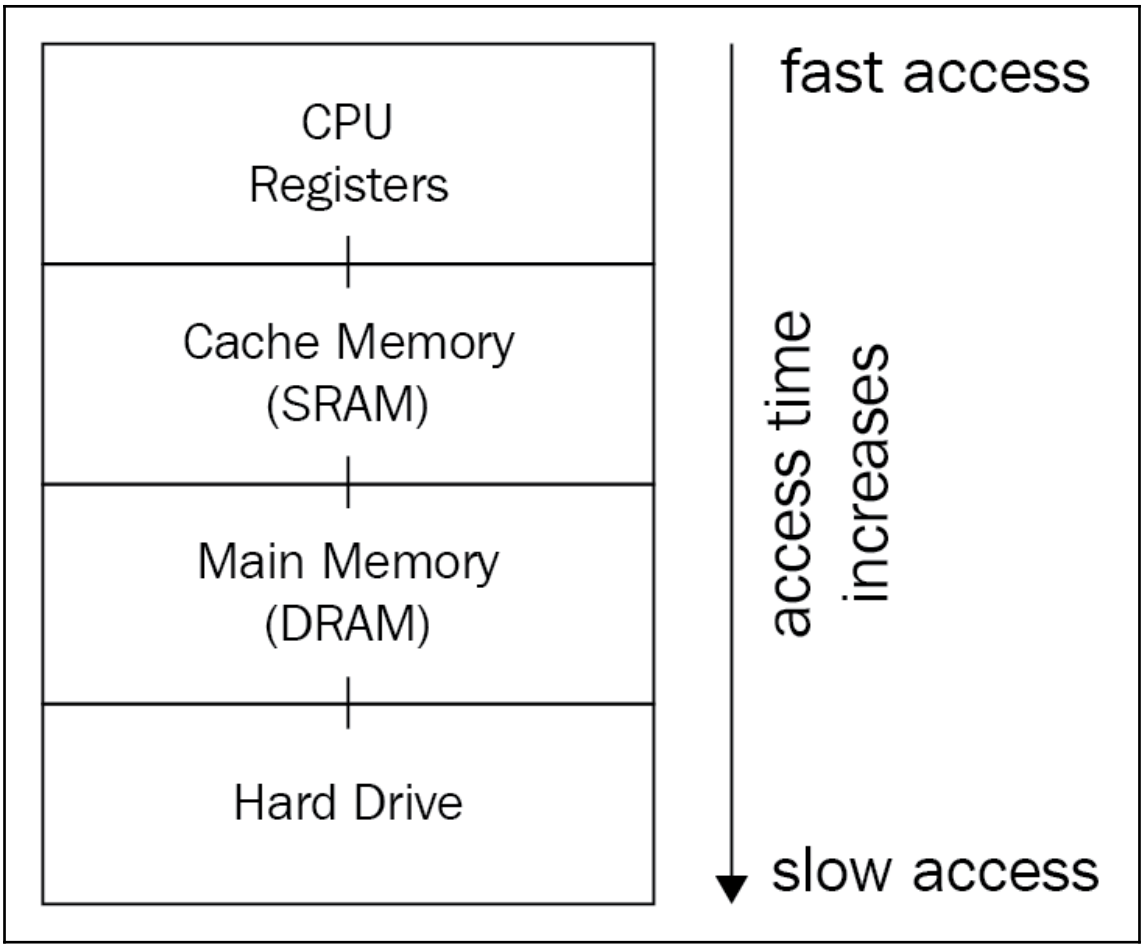
Chapter 5: Memory Management and Smart Pointers

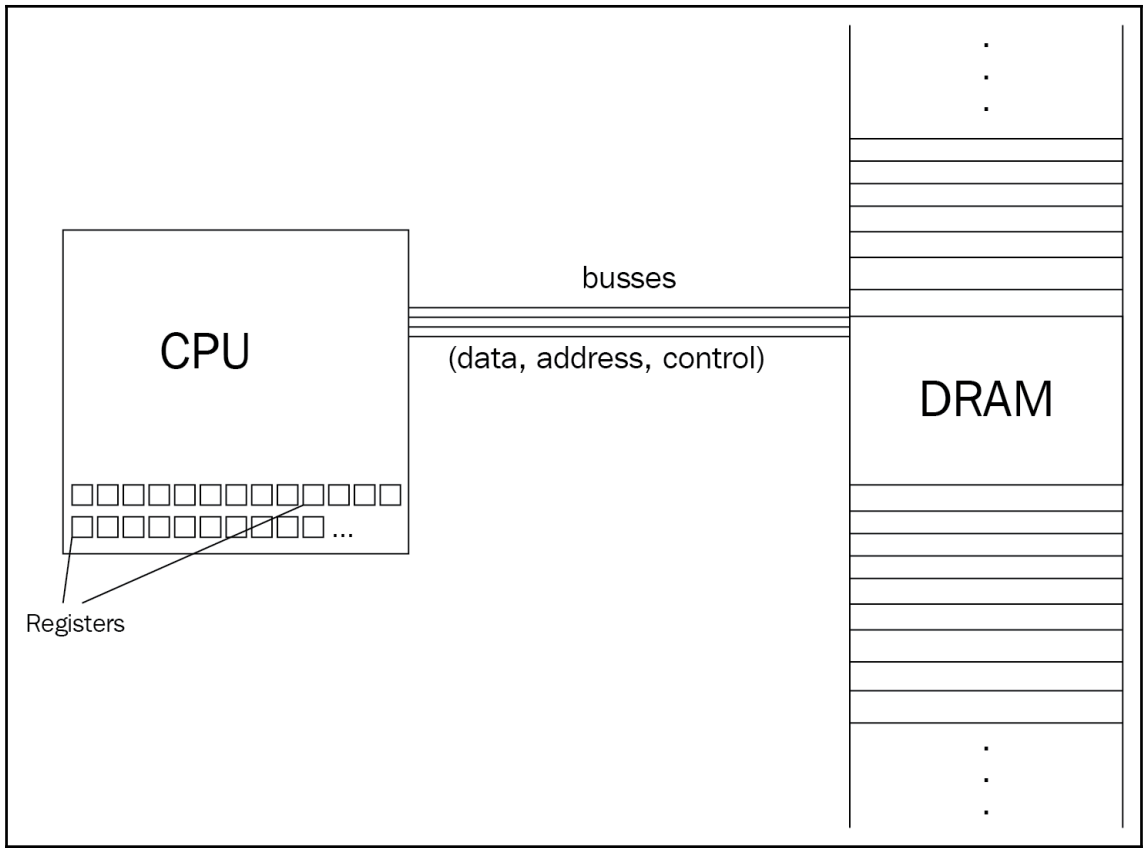


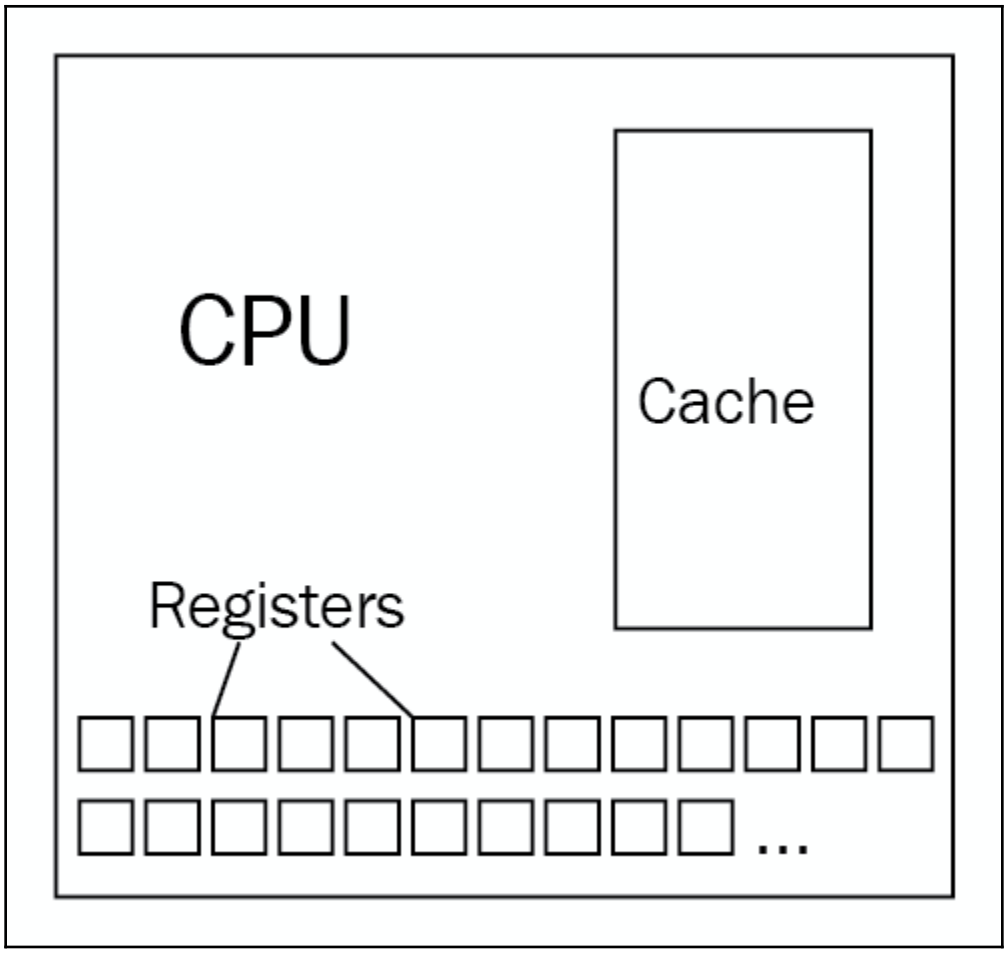


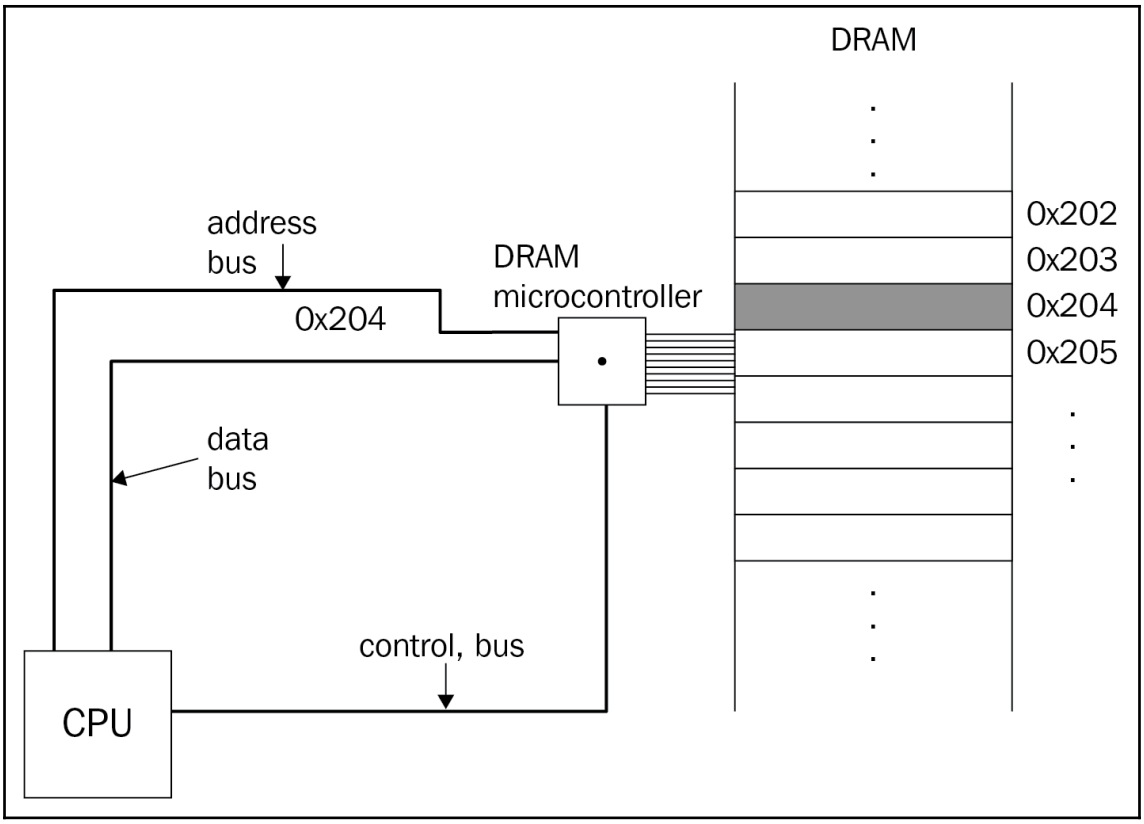

switches are off when relays are on
relay
relay
relay switch

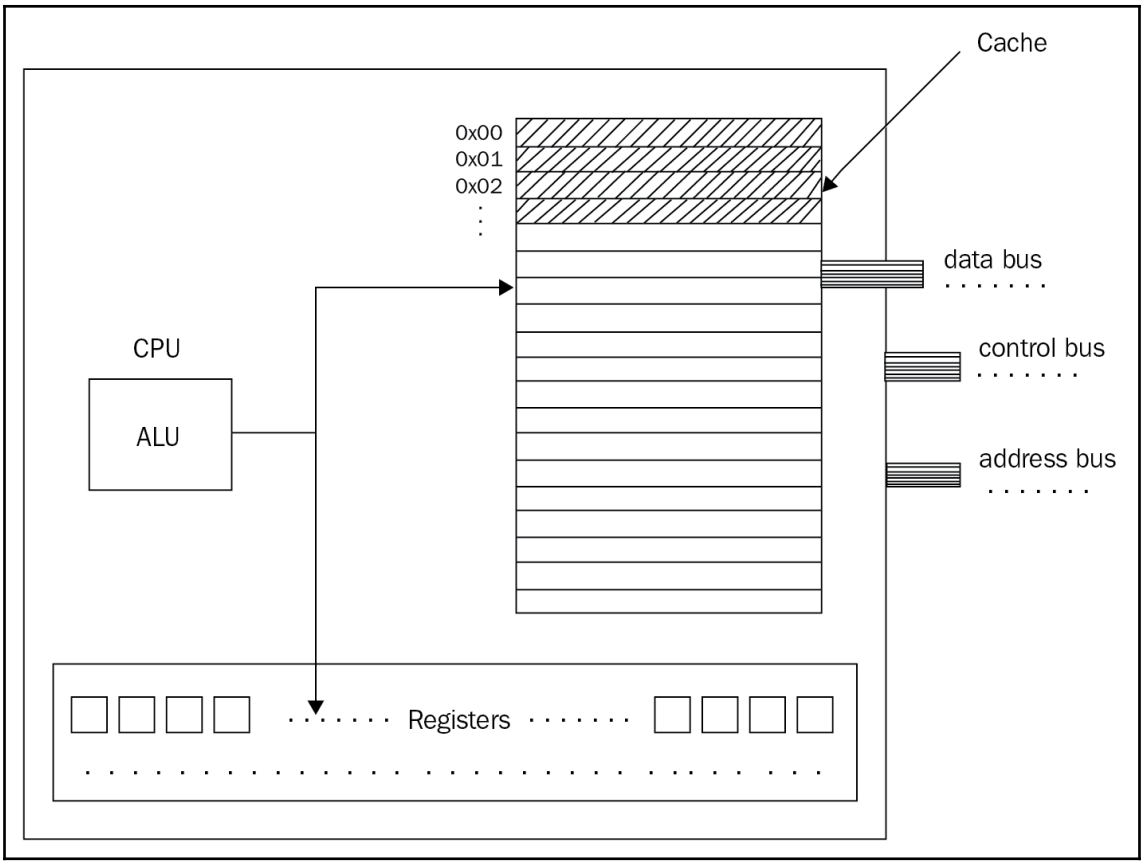

R
Q
S

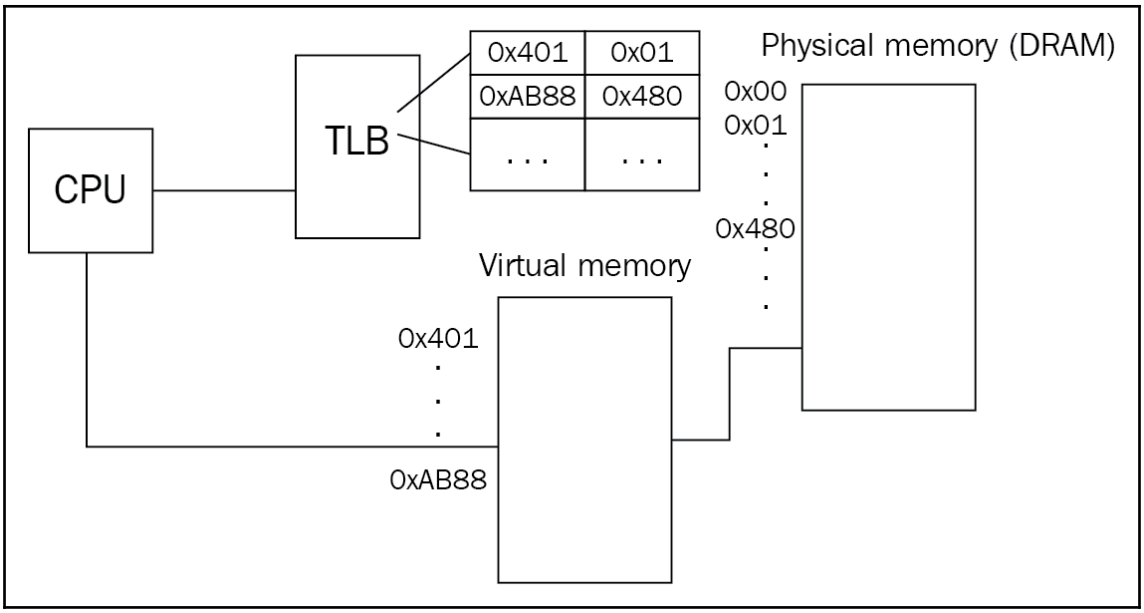

CPU
Registers
Cache Memory
(SRAM)
Main Memory
(DRAM)
Hard Drive
fast access
access time
increases
slow access

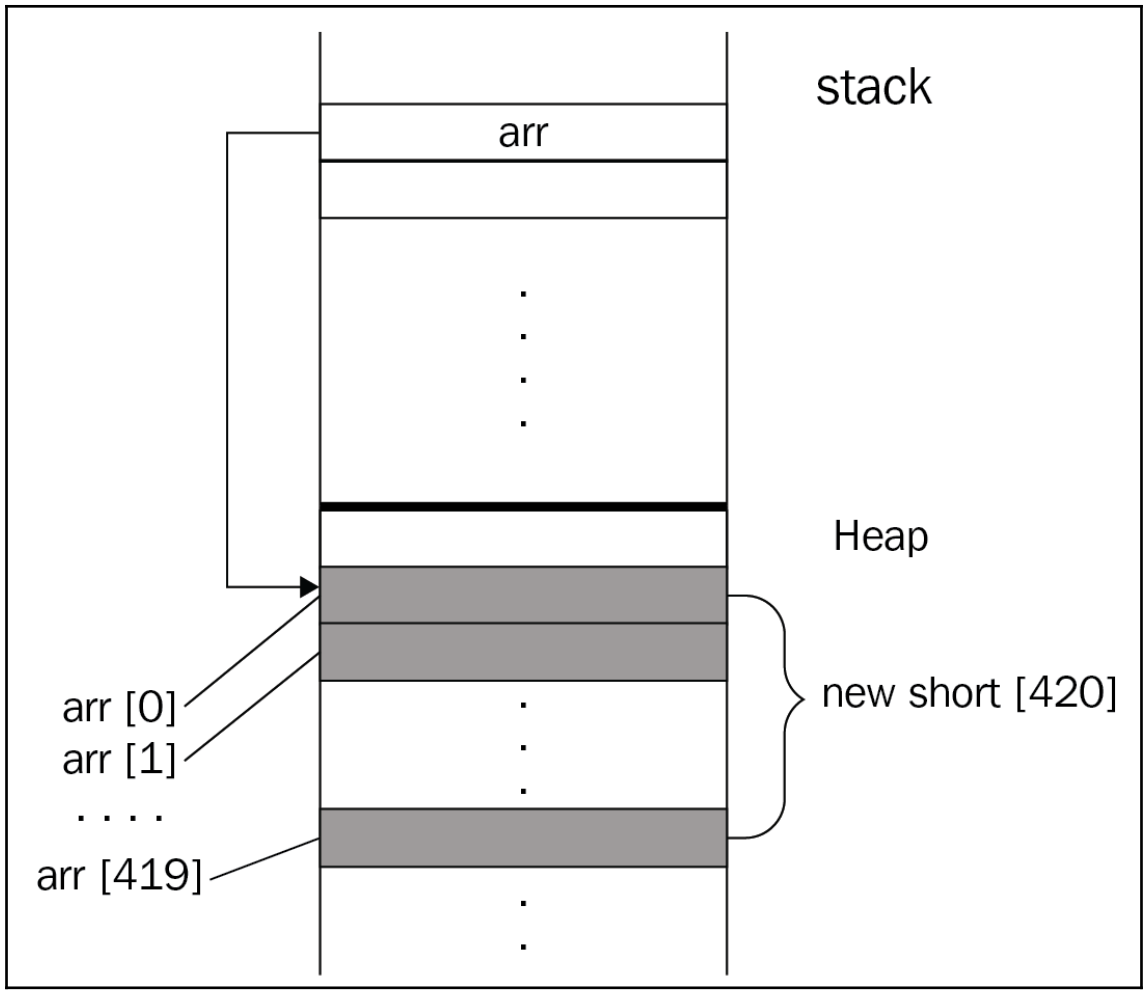

busses
(data, address, control)
CPU
DRAM
Registers

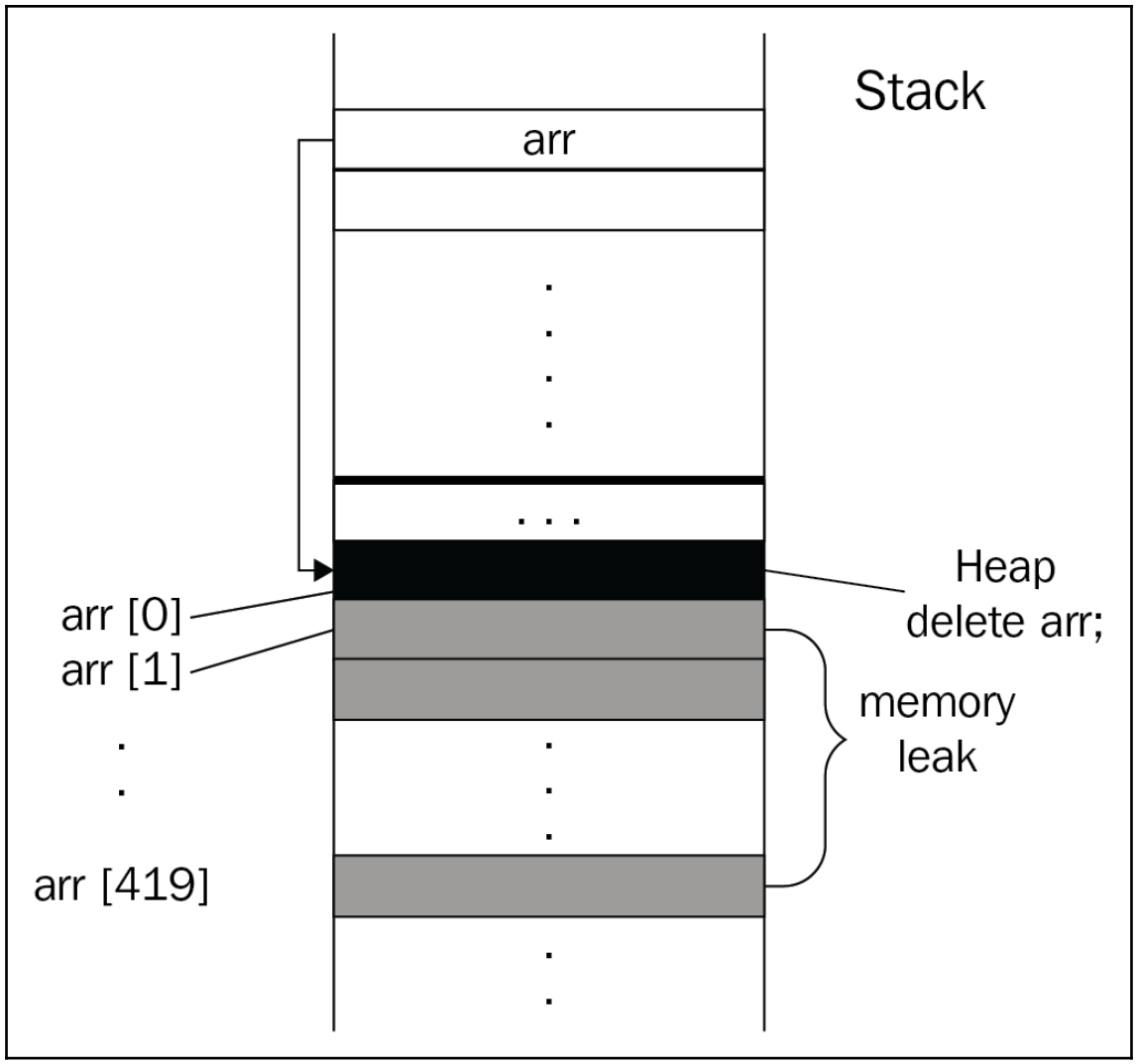

CPU
Cache
Registers
...

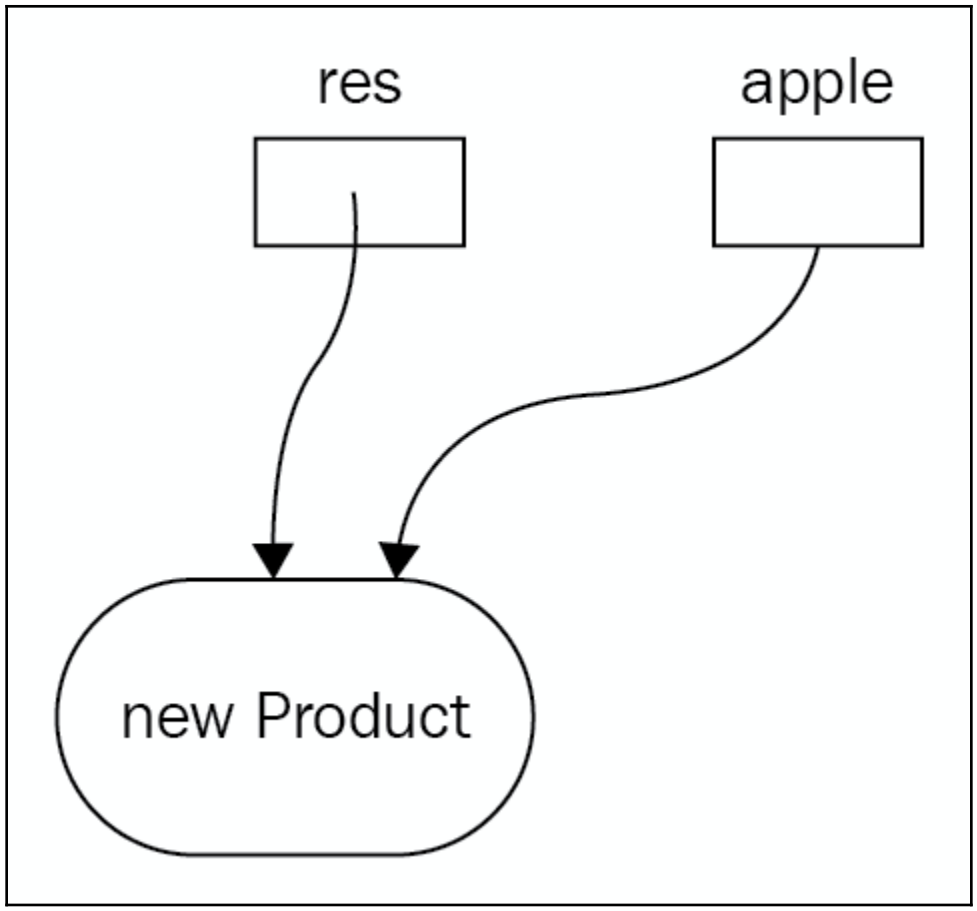

DRAM
address
bus
0x204
DRAM
microcontroller
0x202
0x203
0x204
0x205
data
bus
control, bus
CPU

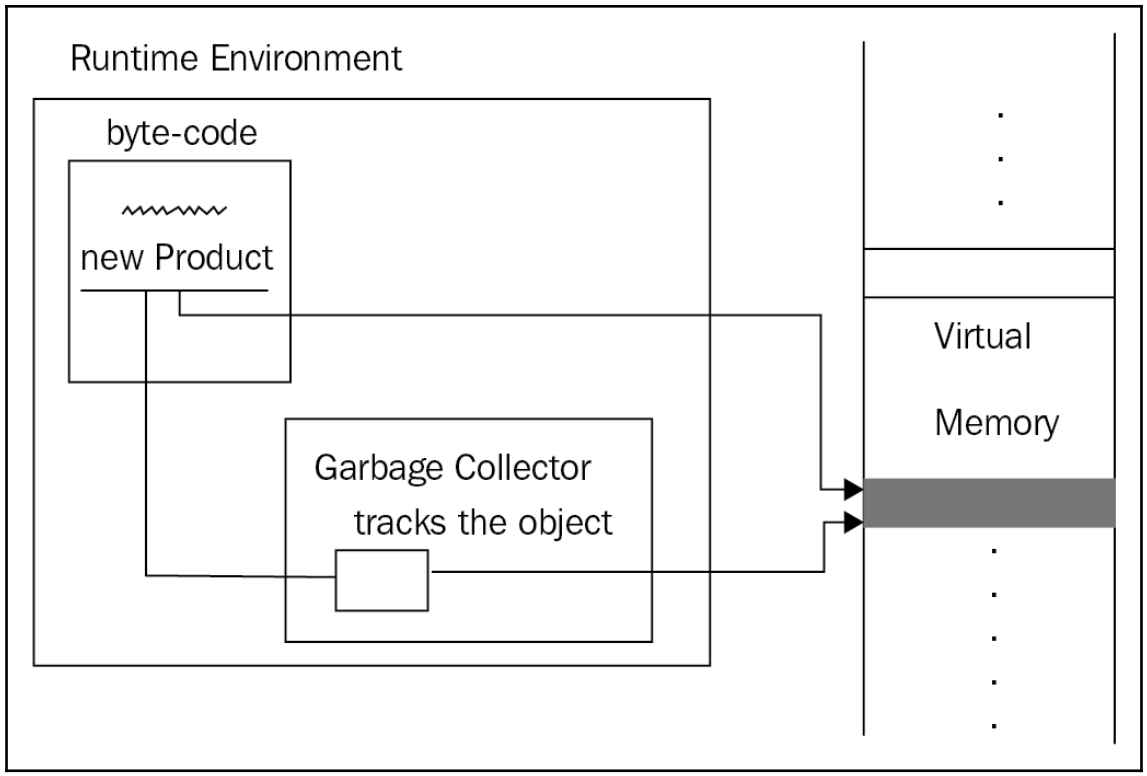

Cache
0x00
0x01
0x02
data bus
CPU
control bus
ALU
address bus
Registers

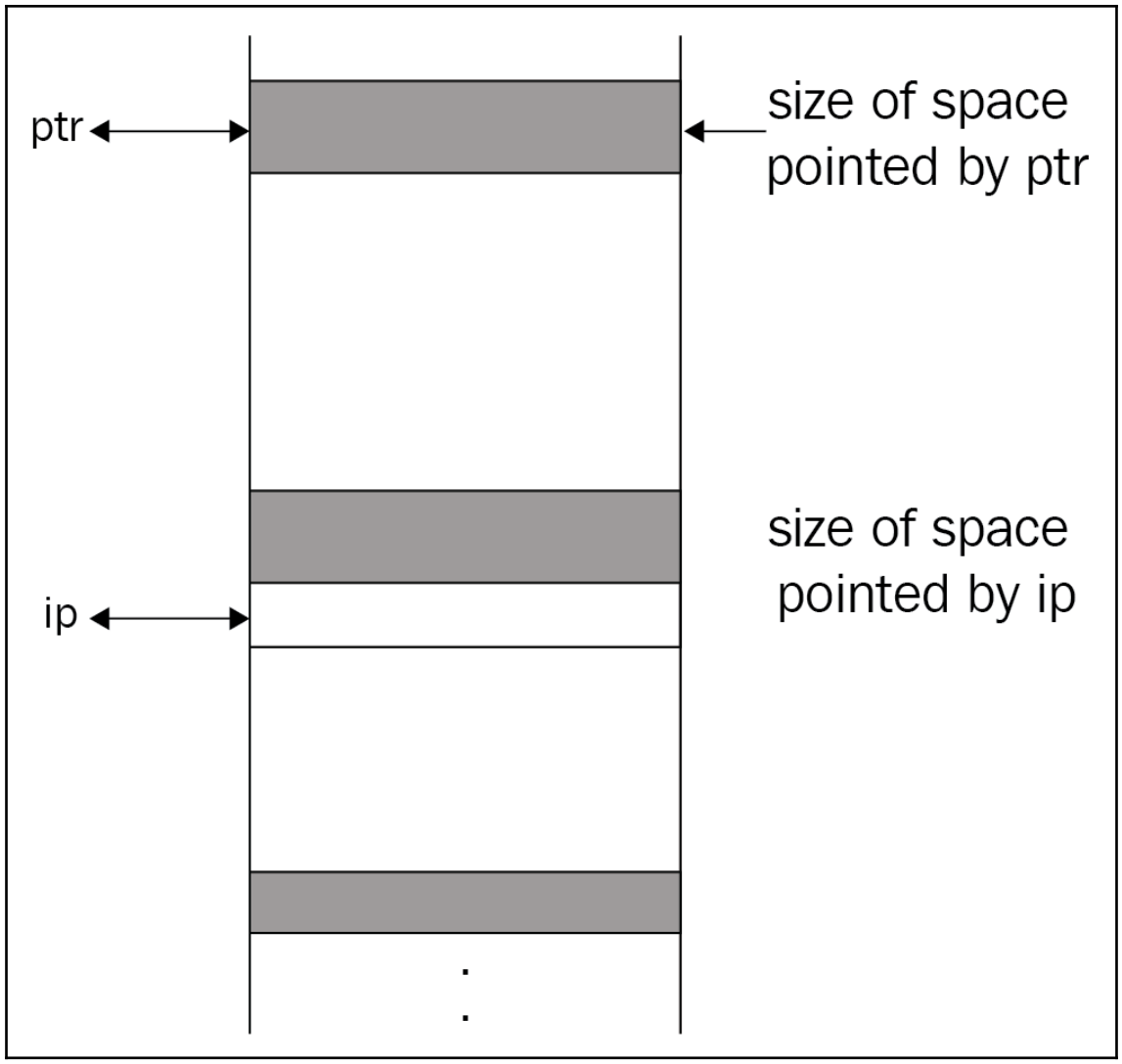

0x401
0x01
0xAB88
0x480
. . .
. . .
TLB
CPU
Physical memory (DRAM)
0x00
0x01
0x480
Virtual memory
0x401
0xAB88


stack
arr
Heap
new short [420]
arr [0]
arr [1]
. . . .
arr [419]

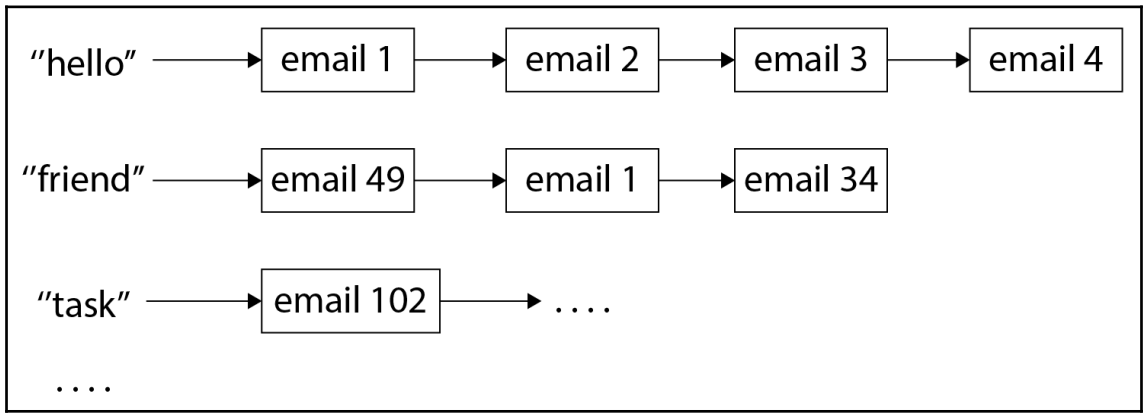

Stack
arr
. . .
Heap
delete arr;
arr [0]
arr [1]
memory
leak
arr [419]

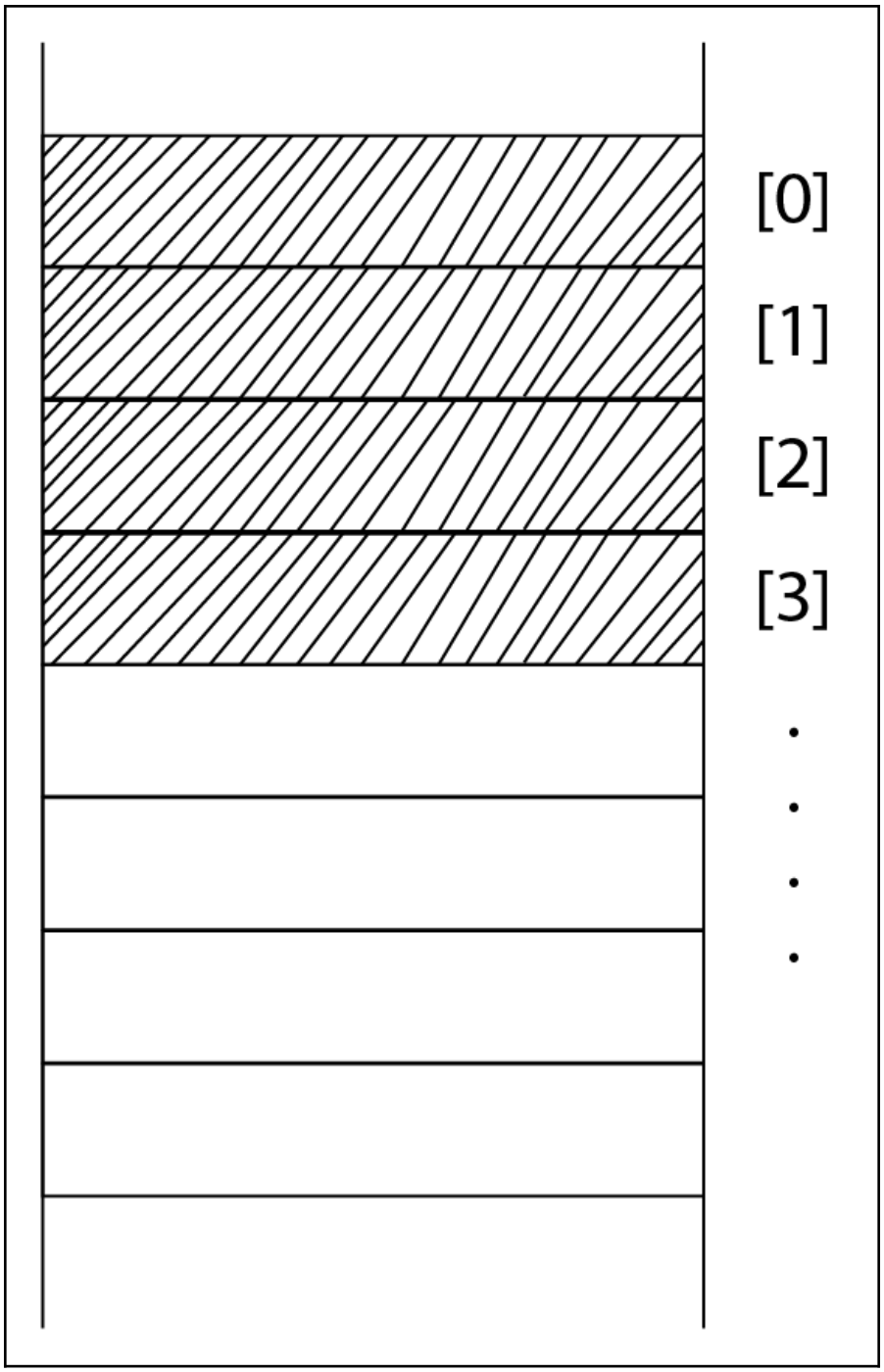

res
apple
new Product

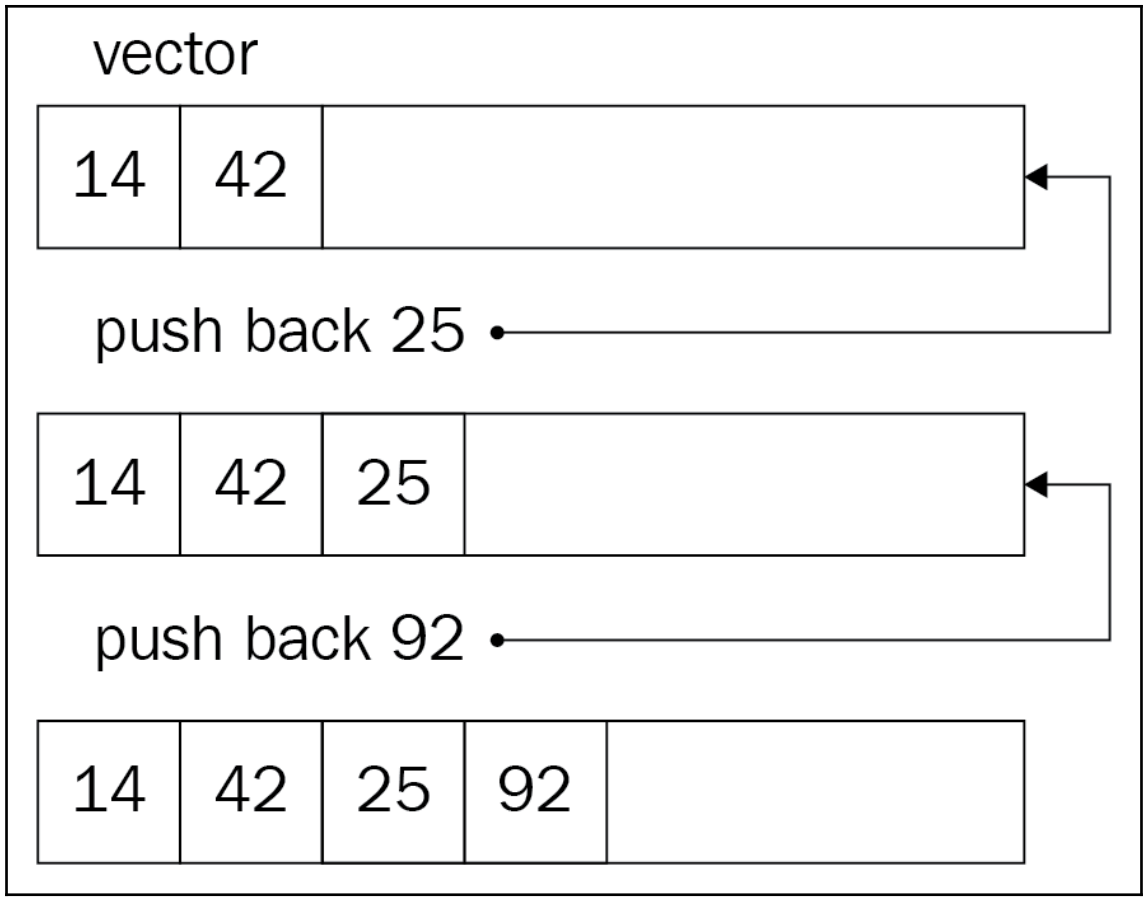

Runtime Environment
byte-code
new Product
Garbage Collector
tracks the object
Virtual
Memory

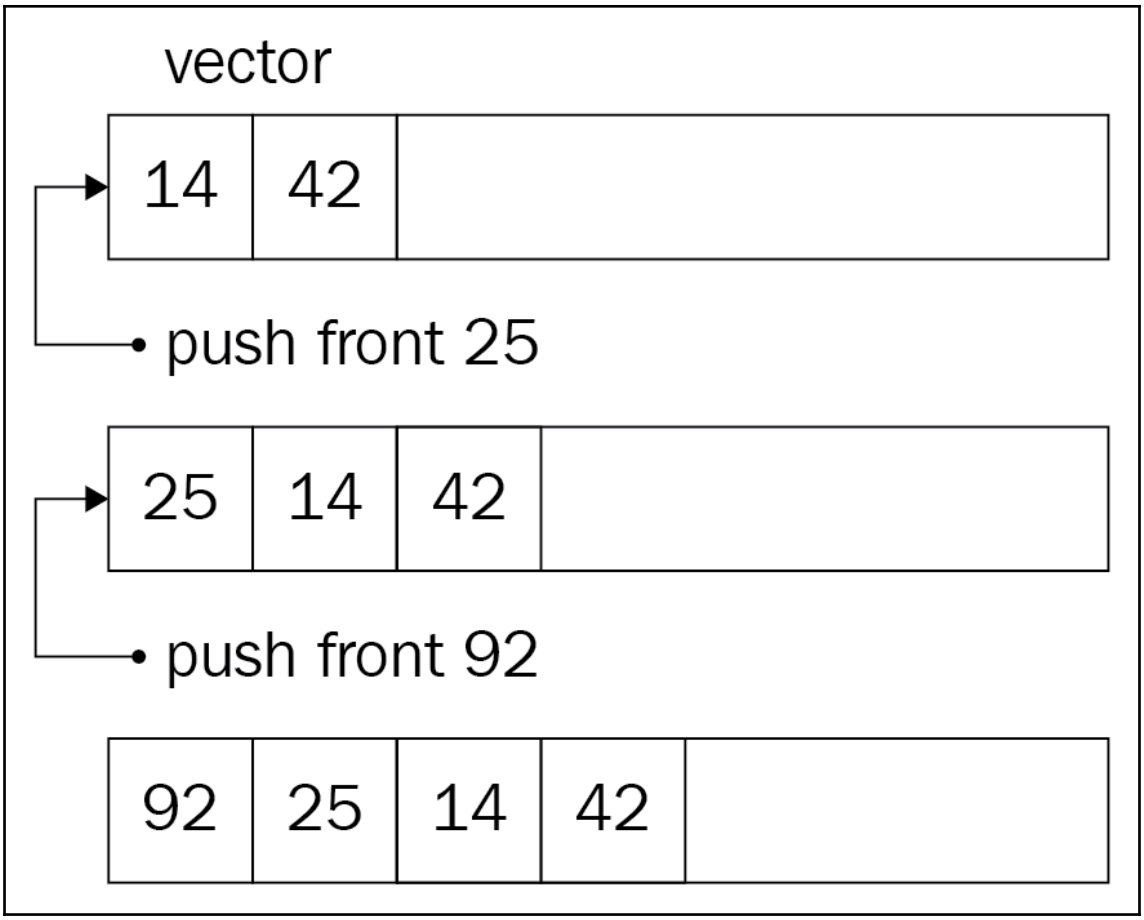

ptr
size of space
pointed by ptr
size of space
pointed by ip
ip
.
.

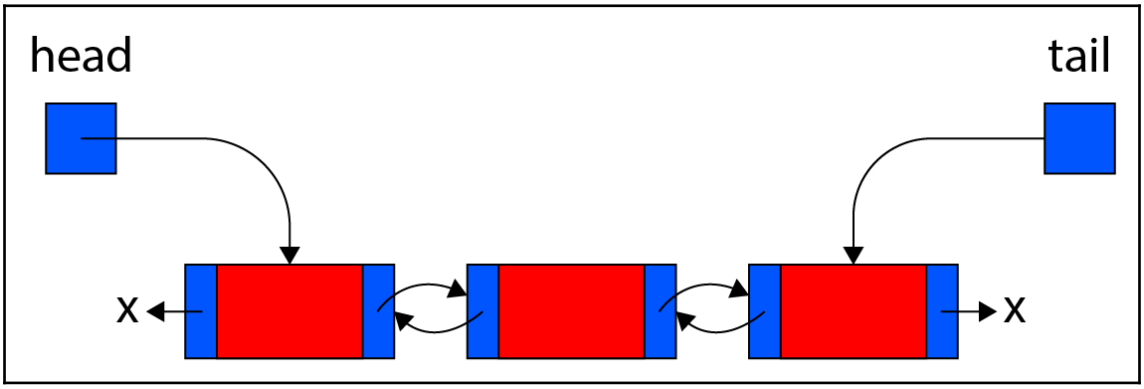
Chapter 6: Digging into Data Structures and Algorithms in STL




[0]
[1]
[2]
[3]


vector
14
42
push back 25
14
42
25
push back 92
14
42
25
92

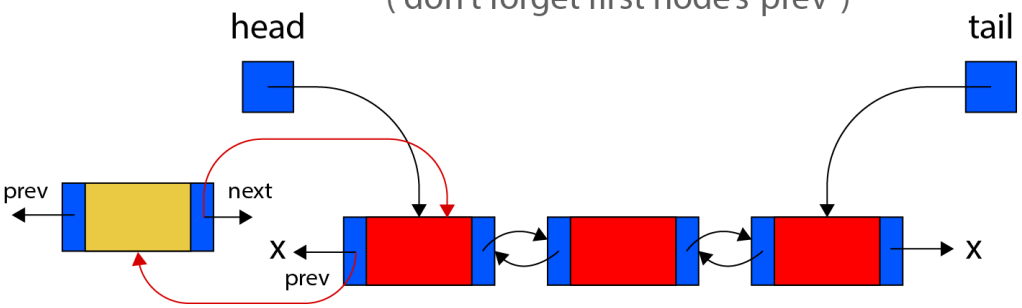

vector
14
42
push front 25
25
14
42
push front 92
92
25
14
42


head
tail
x
x

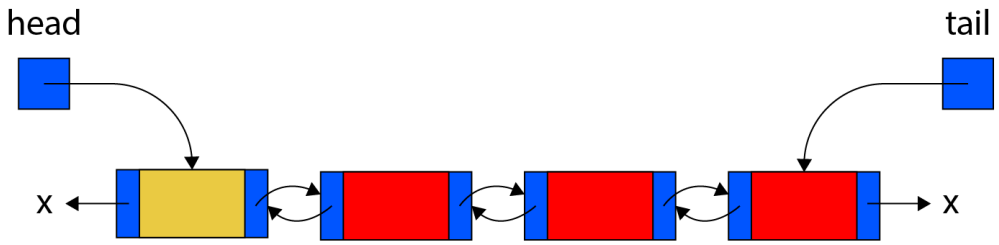
Case A) Insert at the front

step 1) Create node

step 2) Assign its 'next' to the first node (head)
(don't forget first node's 'prev')



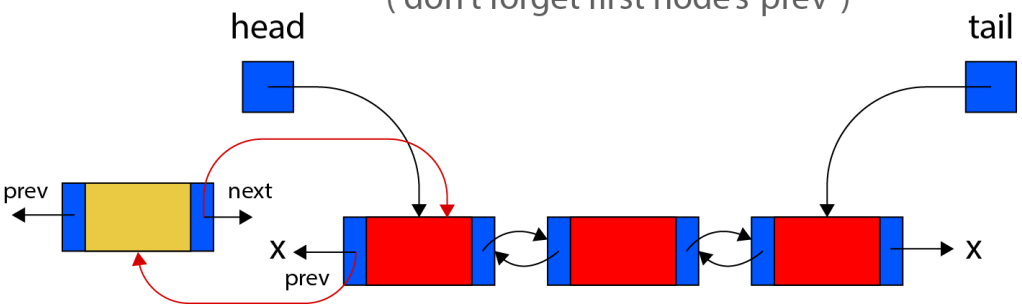
step 3) Make head to point to the new node (new first)



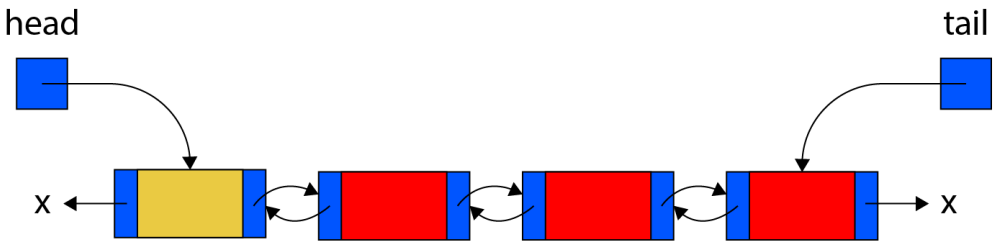
Case A) Insert at the front

step 1) Create node

step 2) Assign its 'next' to the first node (head)
(don't forget first node's 'prev')



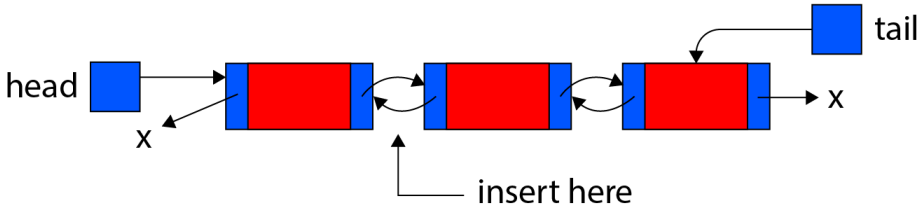
step 3) Make head to point to the new node (new first)



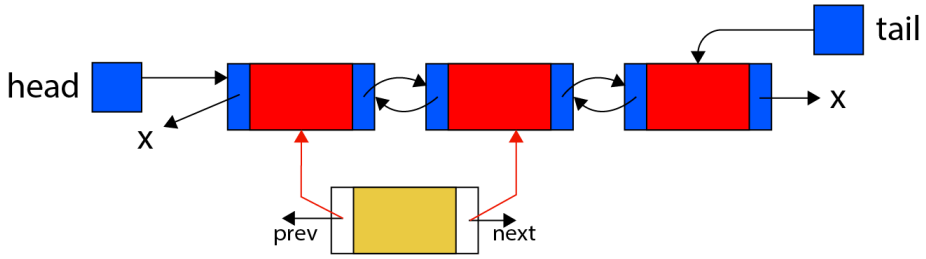
Case C) Insert at middle (any other position)

step 1) Create new node

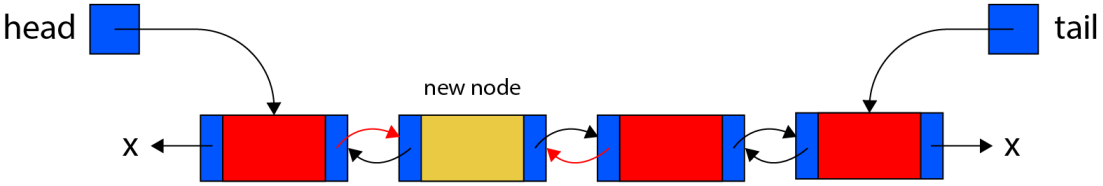
step 2) Find the position to insert

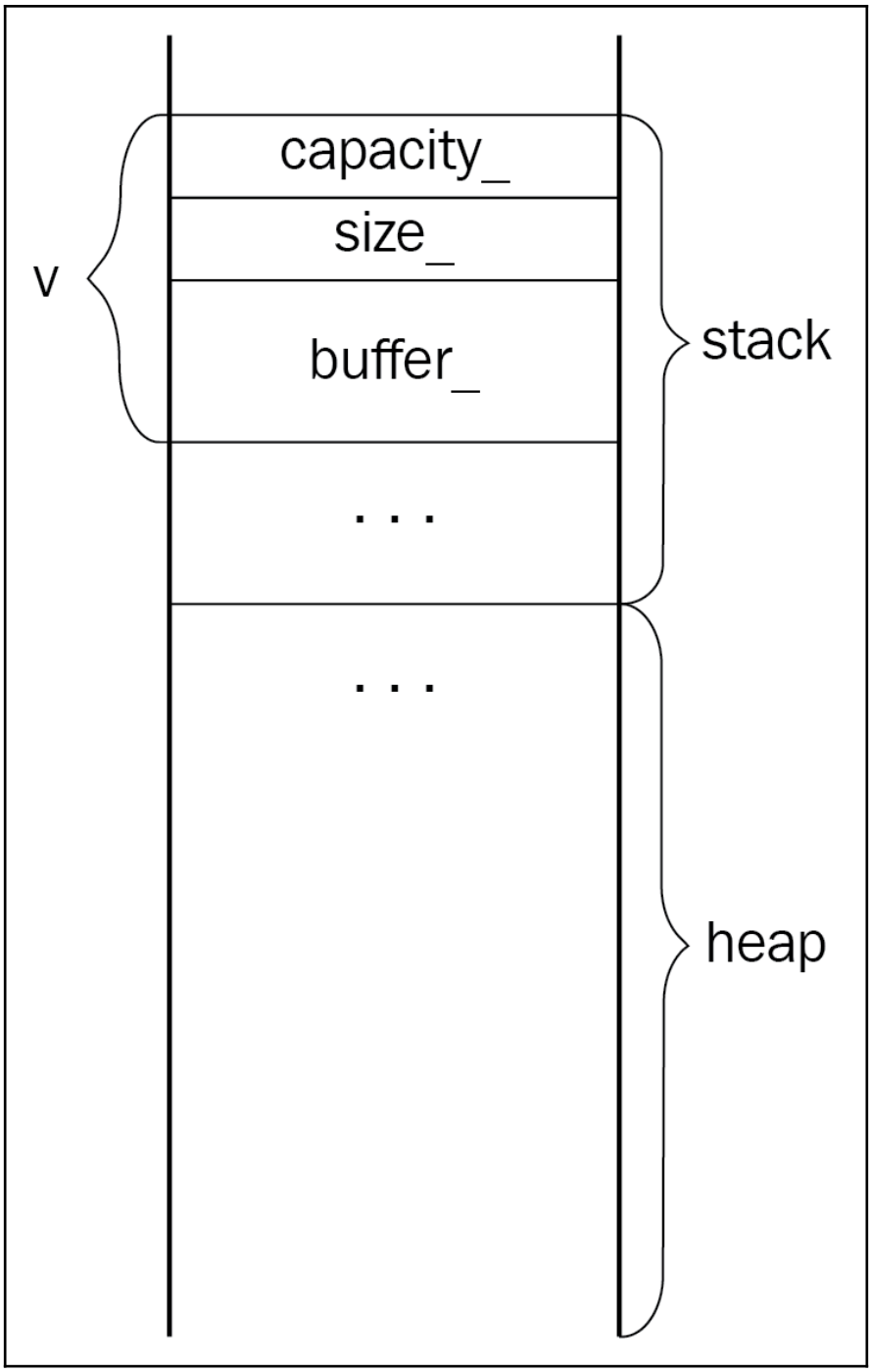


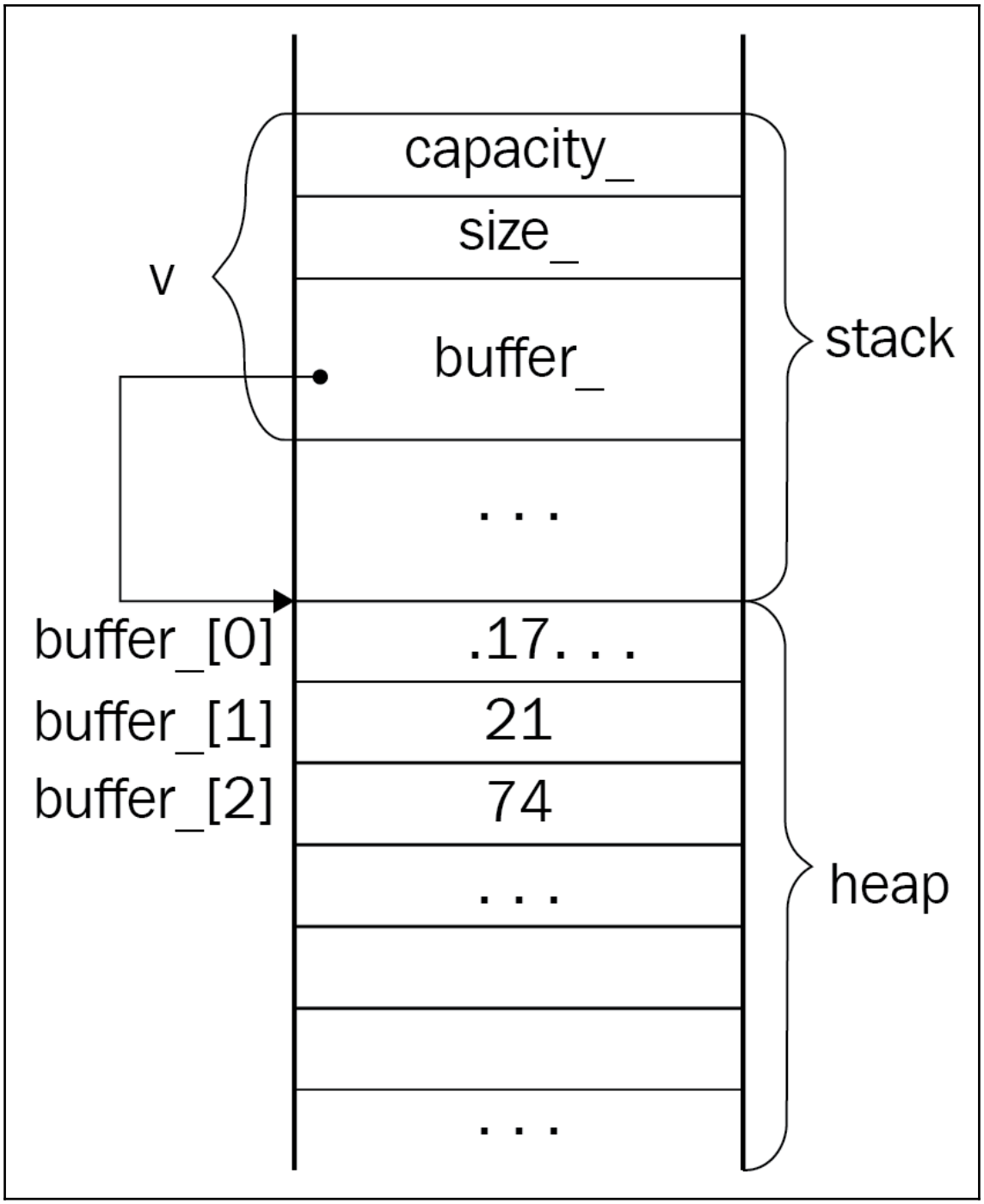
step 3) Assign new node's 'prev' and 'next' proper nodes

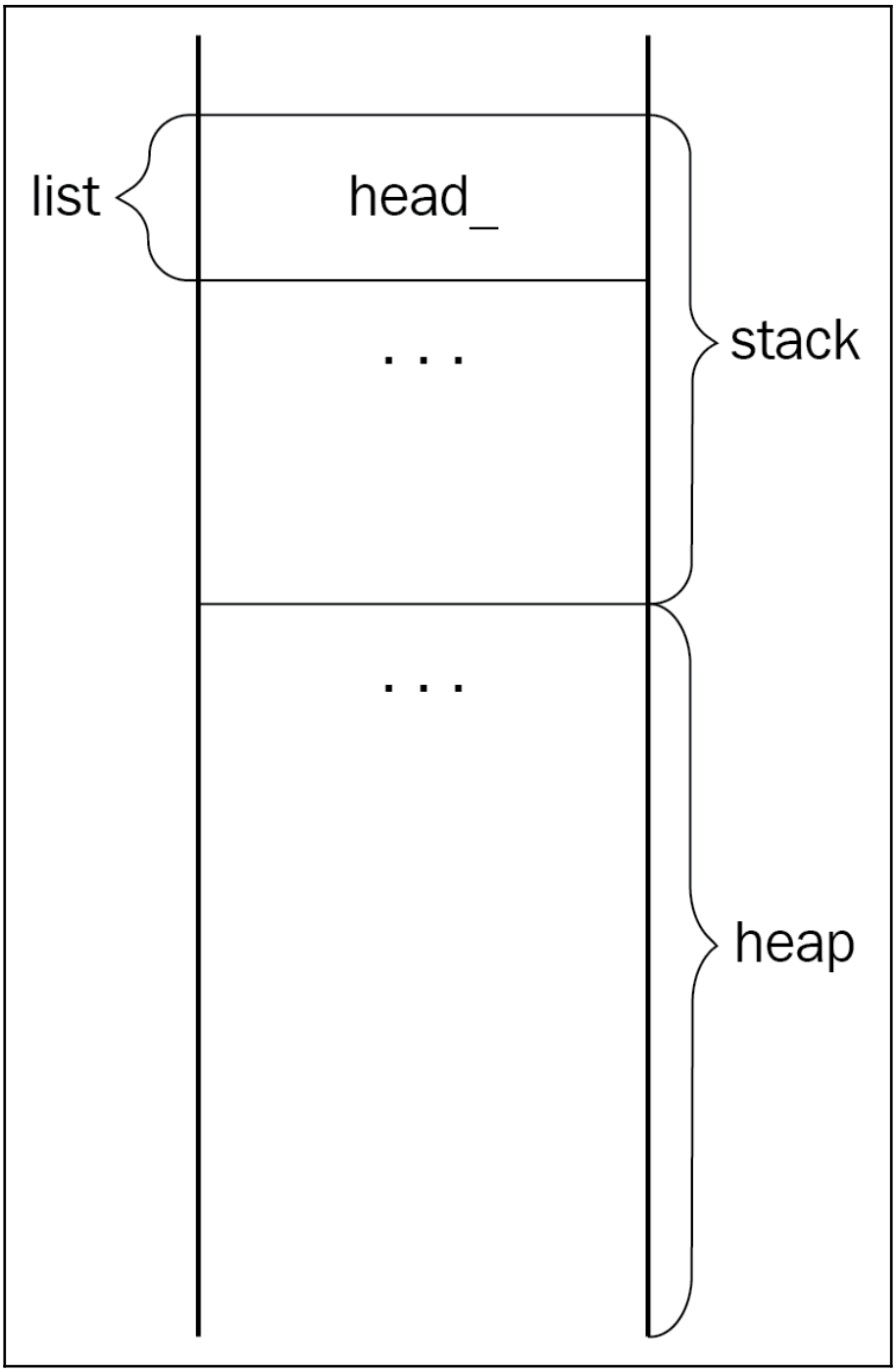


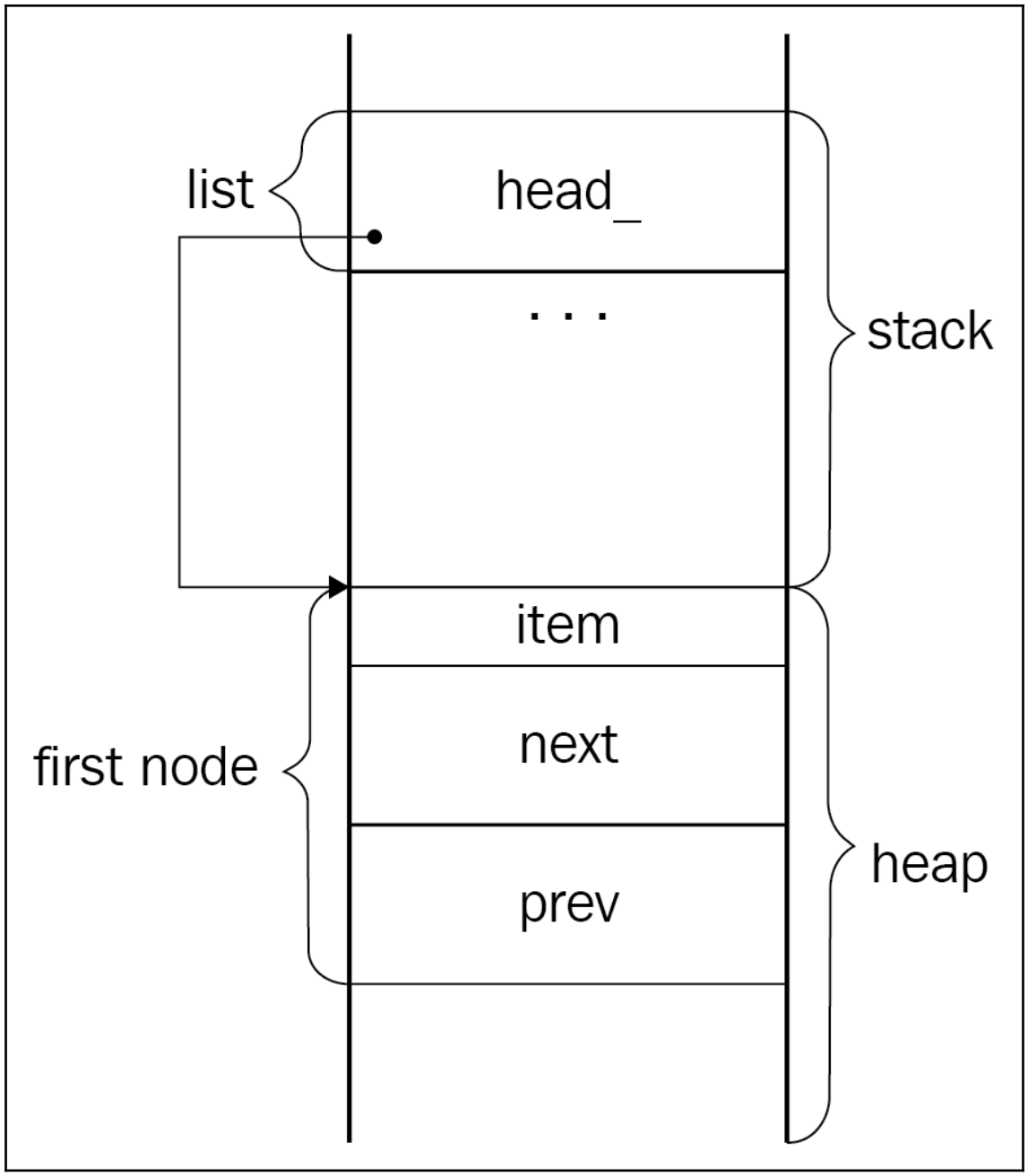
step 4) Update others nodes to point to new node

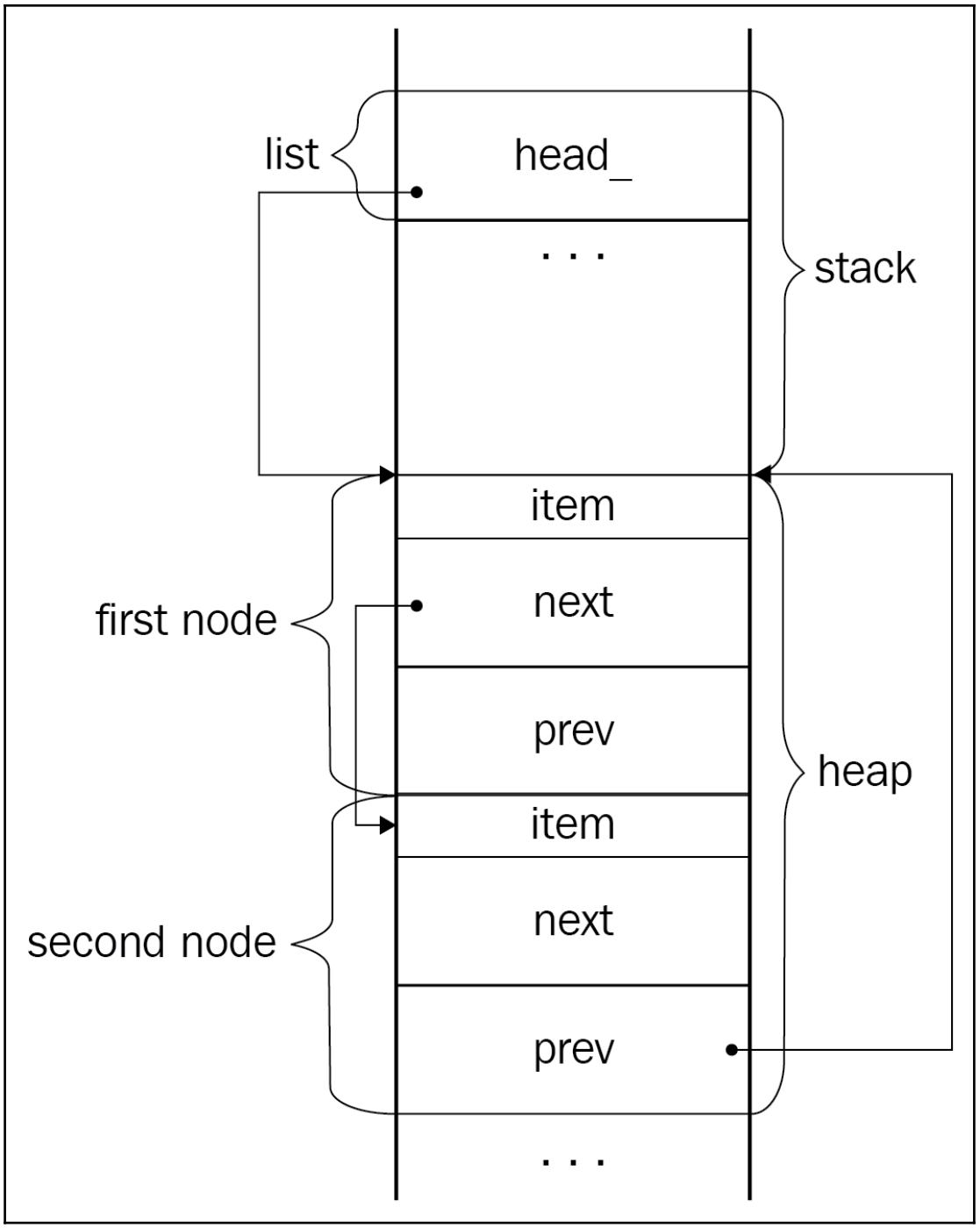


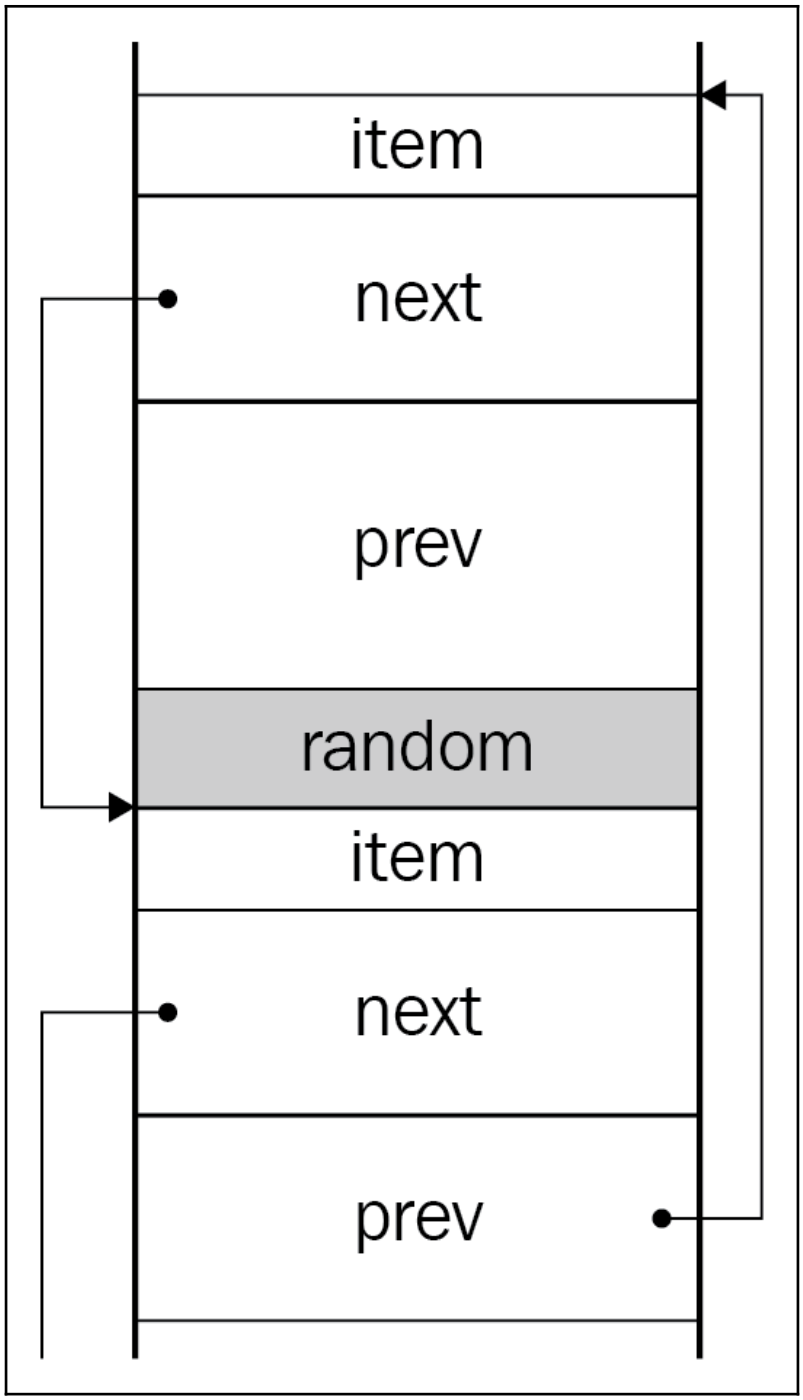

capacity_
size_
v
buffer_
stack
. . .
. . .
heap

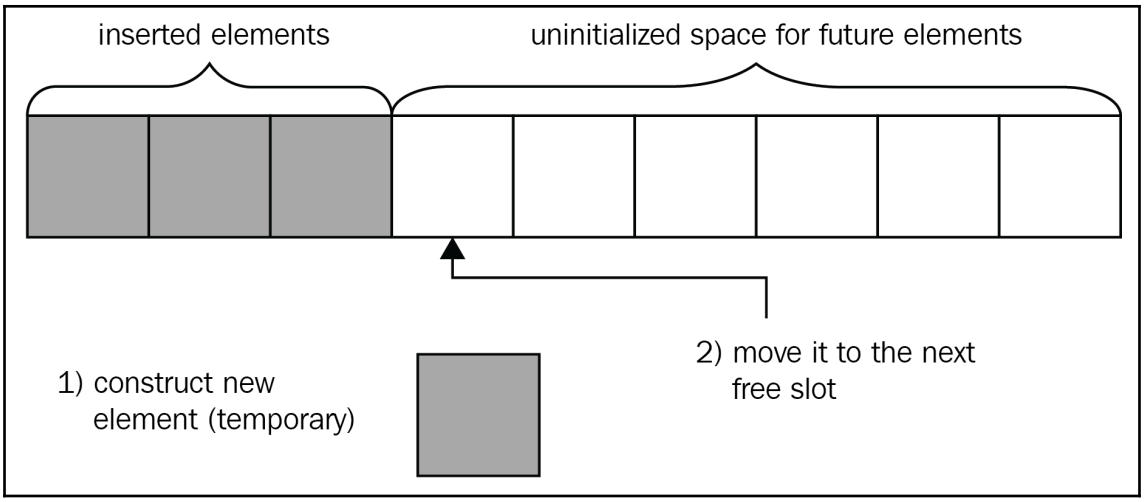

capacity_
size_
v
buffer_
stack
. . .
buffer_[0]
.17. . .
buffer_[1]
21
buffer_[2]
74
. . .
heap
. . .

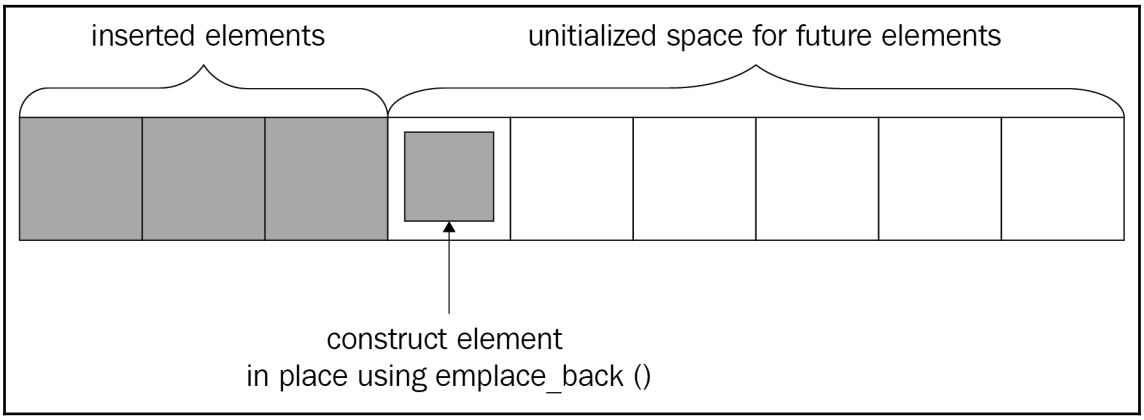

list
head_
. . .
stack
. . .
heap

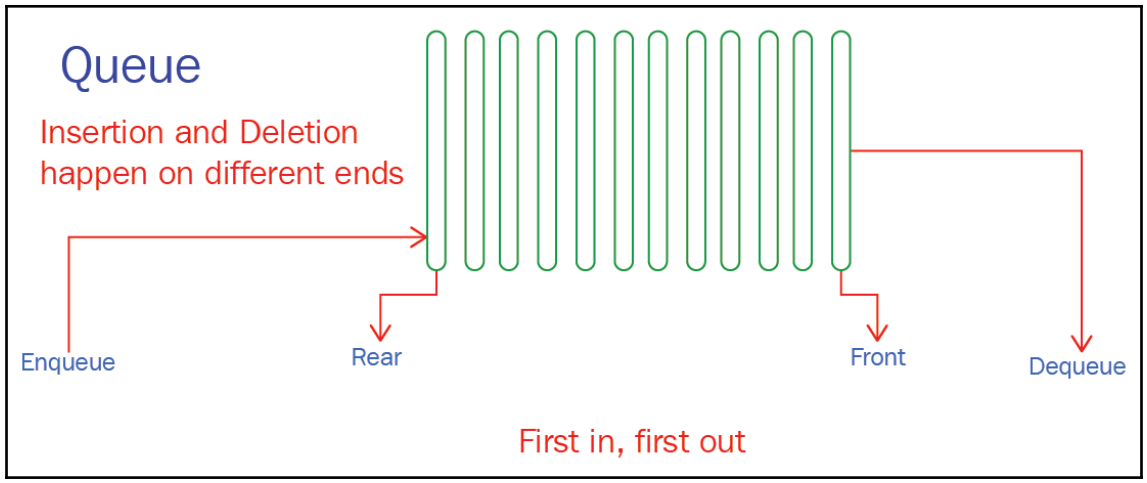

list
head_
. . .
stack
item
next
first node
prev
heap

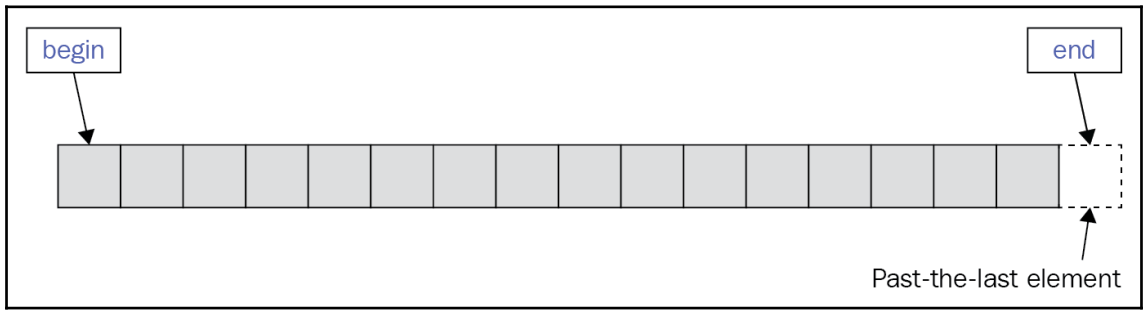

list
head_
. . .
stack
item
first node
next
prev
heap
item
second node
next
prev
. . .

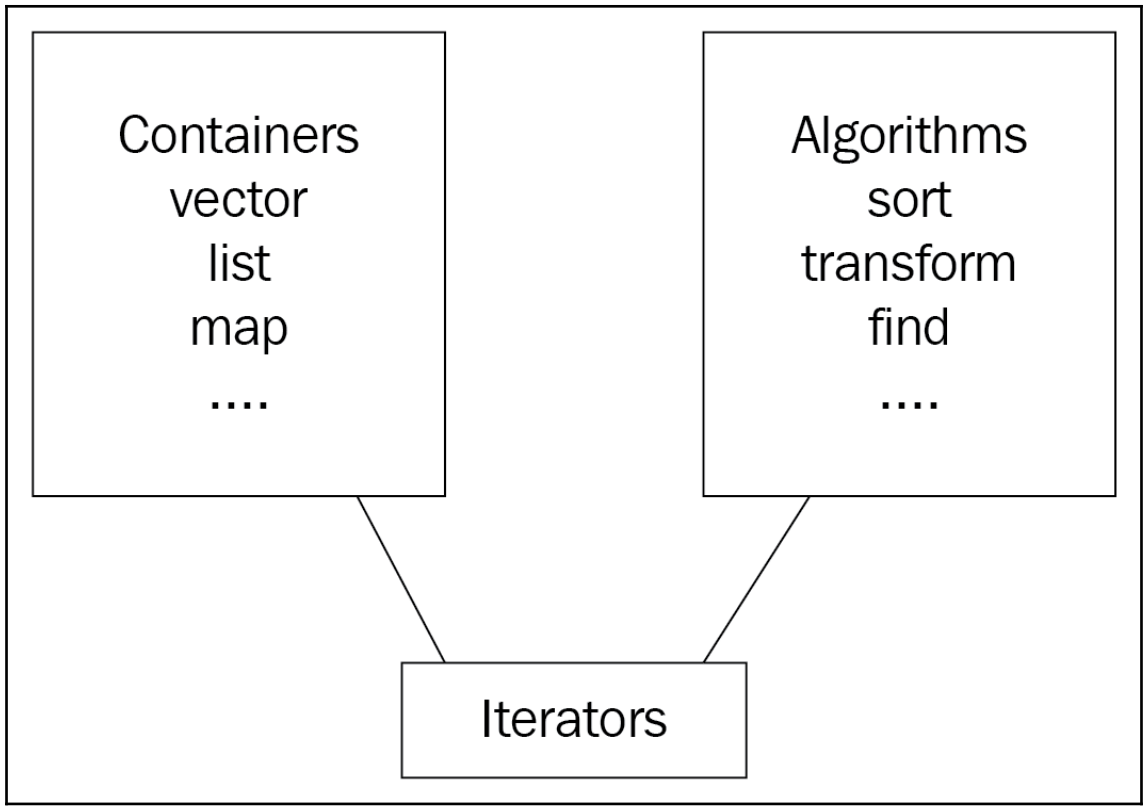

item
next
prev
random
item
next
prev

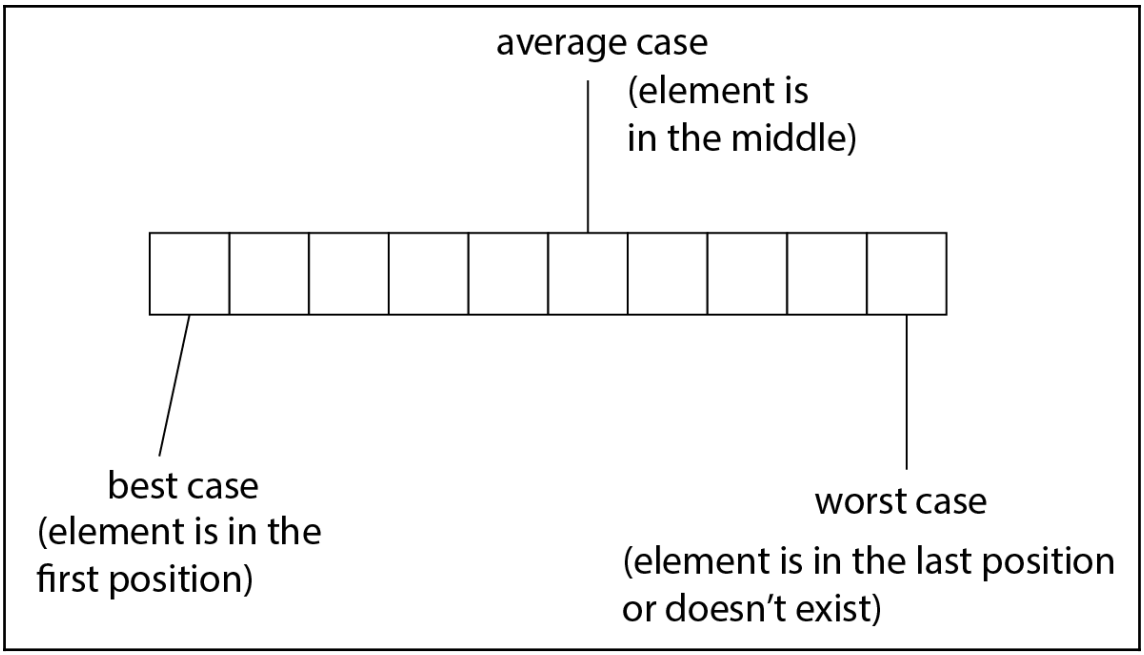

inserted elements
uninitialized space for future elements
1) construct new
element (temporary)
2) move it to the next
free slot

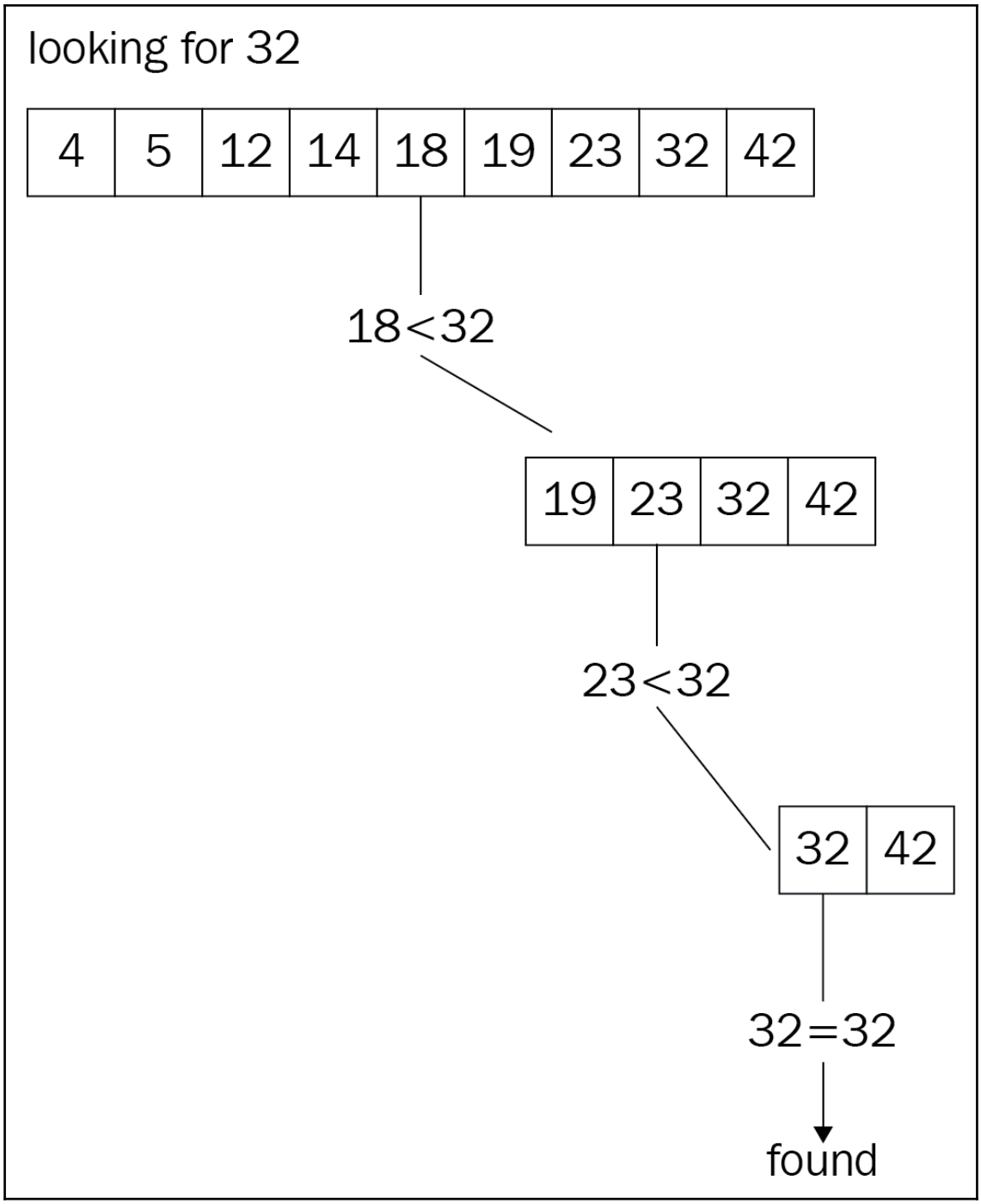

inserted elements
unitialized space for future elements
construct element
in place using emplace_back ()

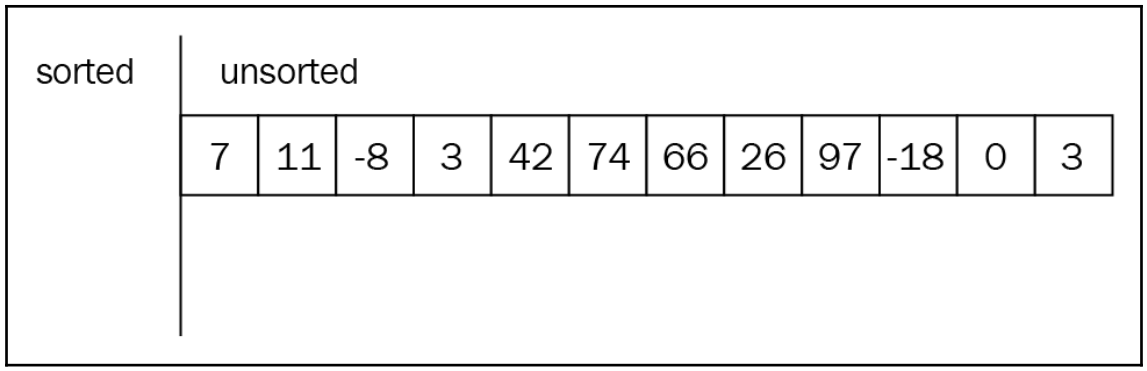

Queue
Insertion and Deletion
happen on different ends
Enqueue
Rear
Front
Dequeue
First in, first out

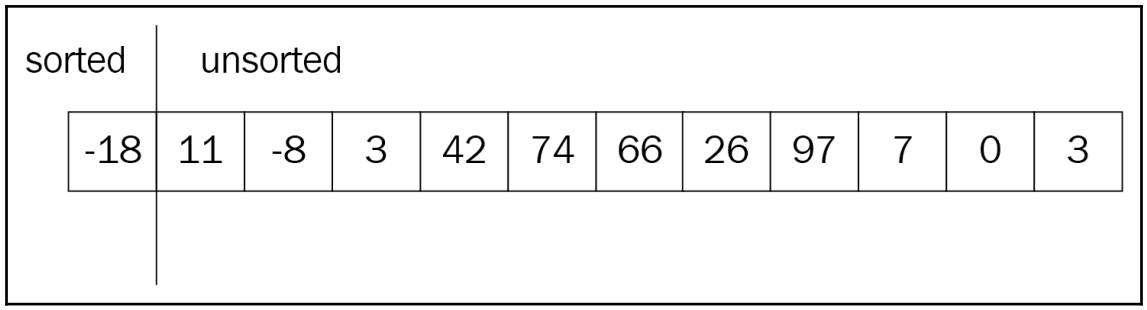

begin
end
Past-the-last element

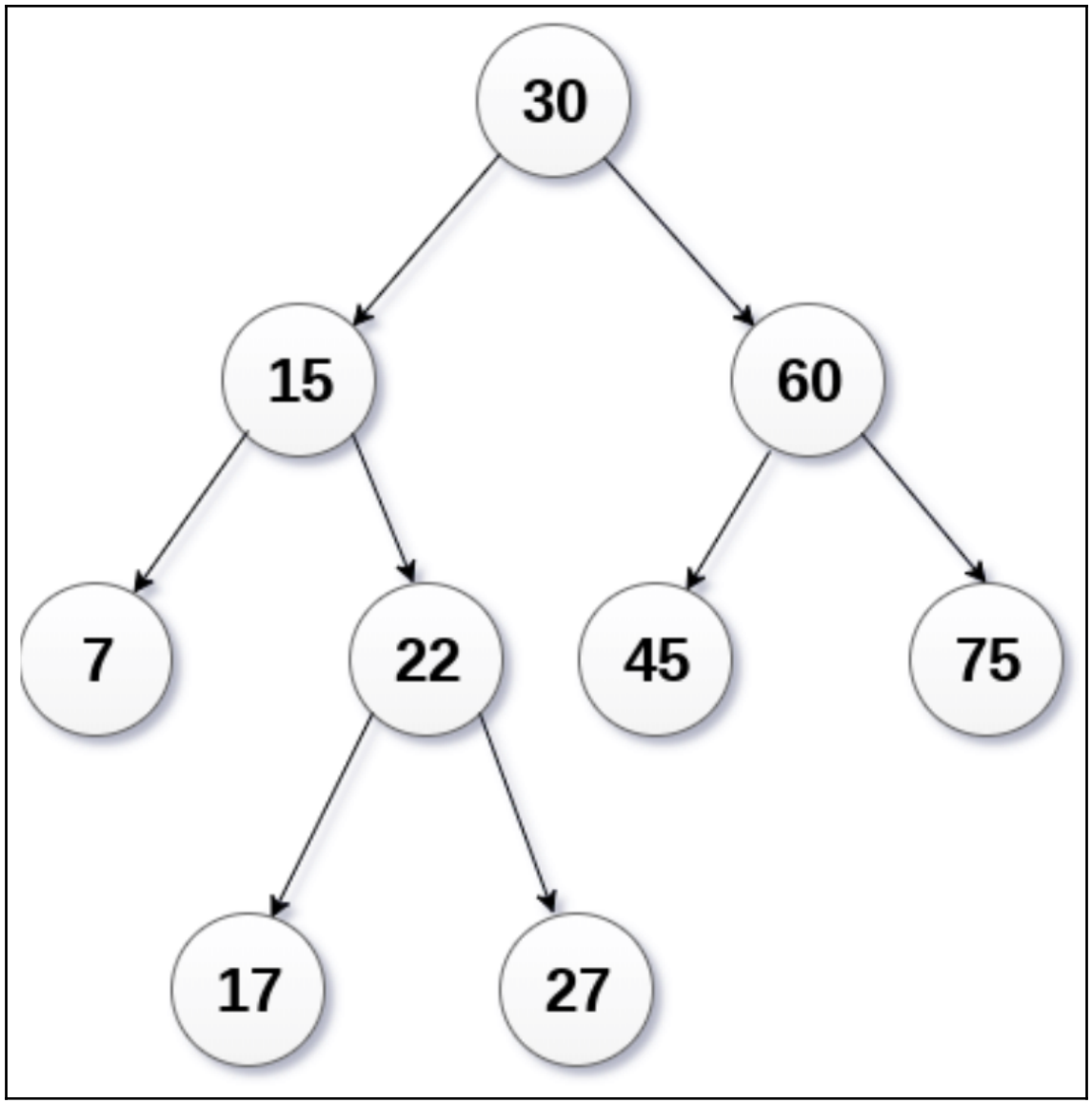

Containers
vector
list
map
....
Algorithms
sort
transform
find
....
Iterators

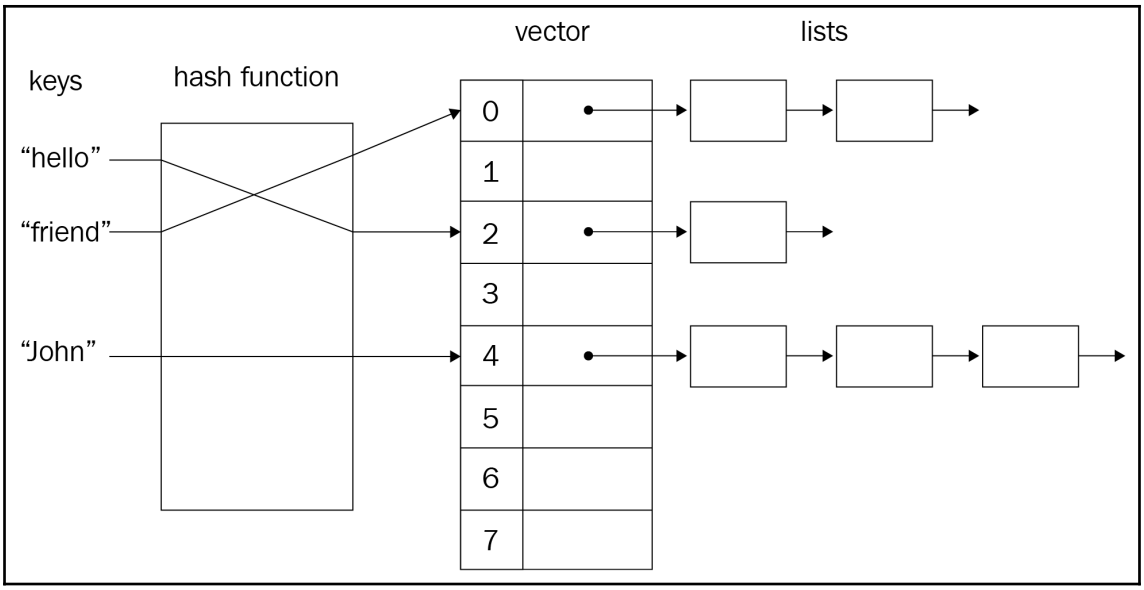

average case
(element is
in the middle)
best case
(element is in the
first position)
worst case
(element is in the last position
or doesn't exist)

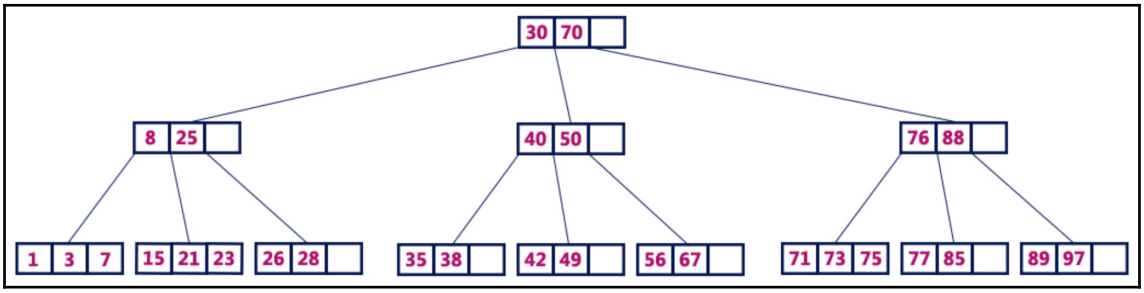

looking for 32
4
5
12
14
18
19
23
32
42
18<32
19
23
32
42
23<32
32
42
32=32
found

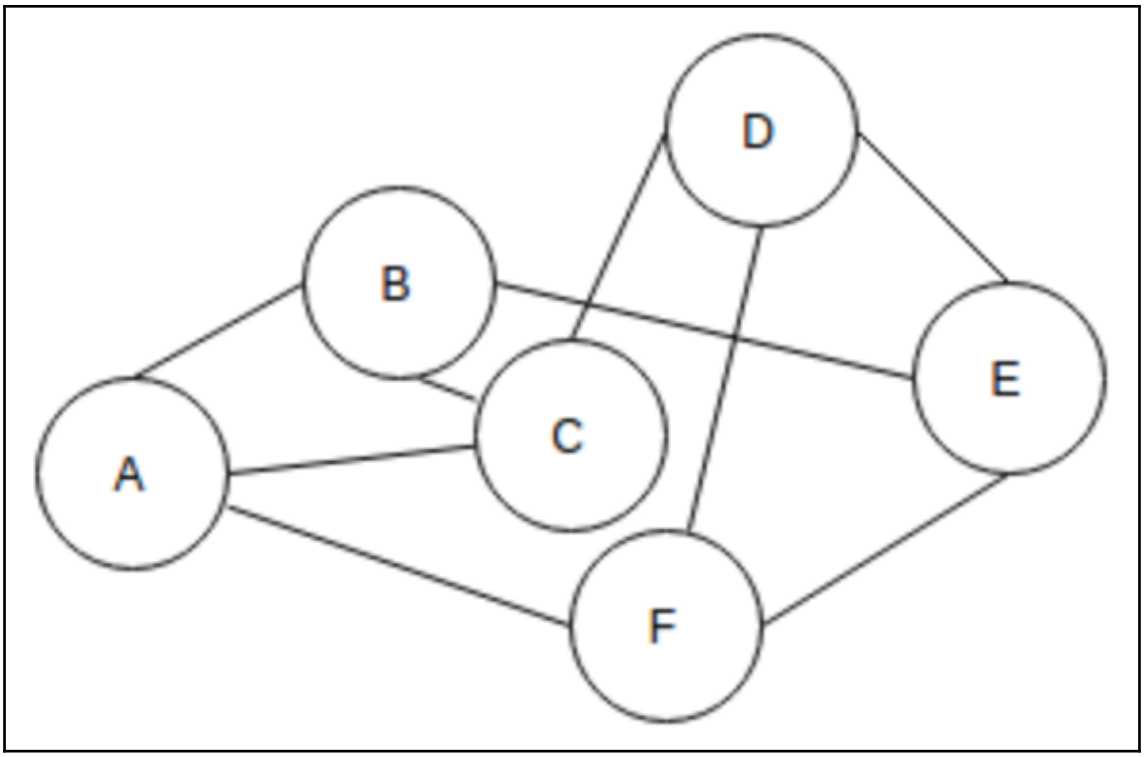

sorted
unsorted
7
11
-8
3
42
74
66
26
97
-18
0
3

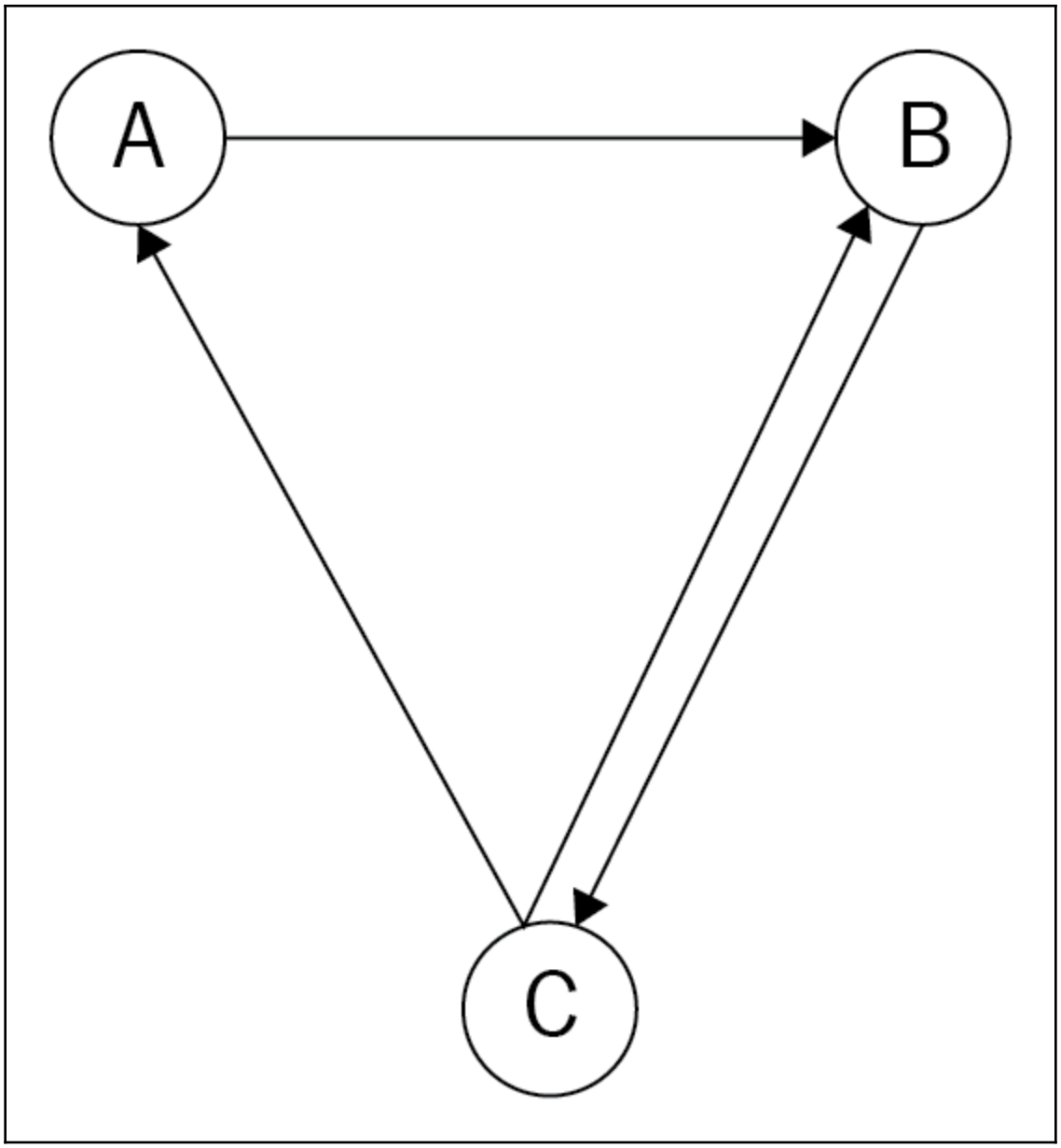

sorted
unsorted
-18
11
-8
3
42
74
66
26
97
7
0
3

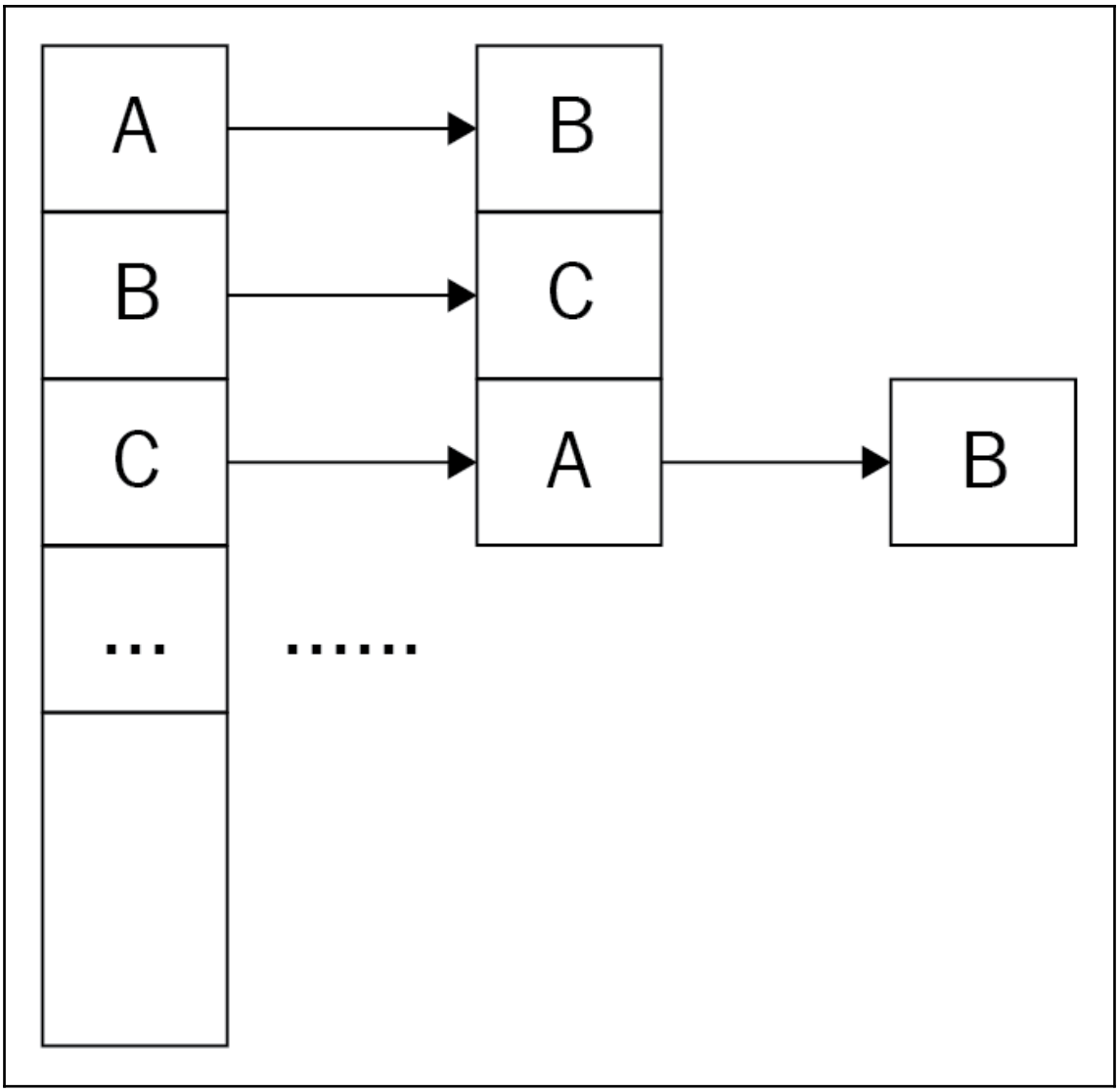

30
15
60
7
22
45
75
17
27


vector
lists
keys
hash function
“hello”
“friend”
“John”
0
1
2
3
4
5
6
7


30 70
8 25
40 50
76 88
1 3 7
15 21 23
26 28
35 38
42 49
56 67
71 73 75
77 85
89 97

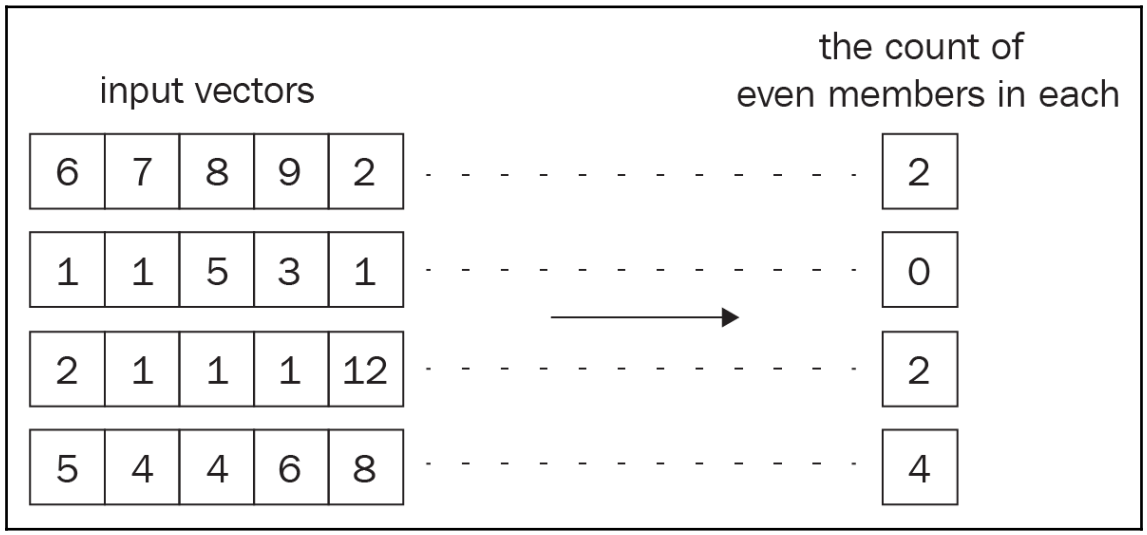

D
B
E
C
A
F

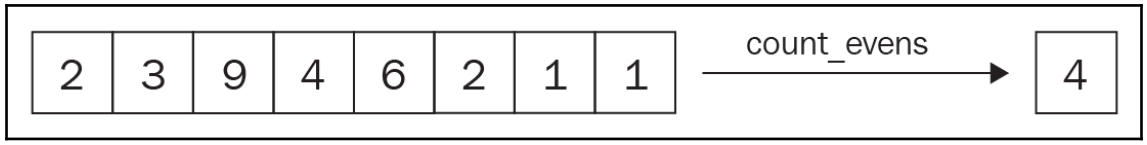

A
B
C

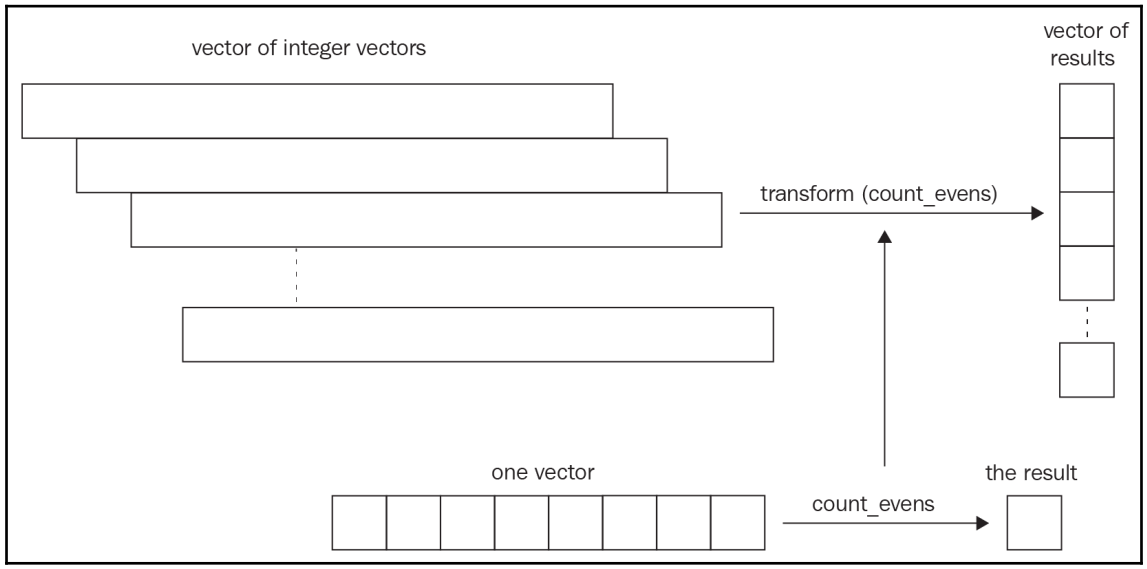


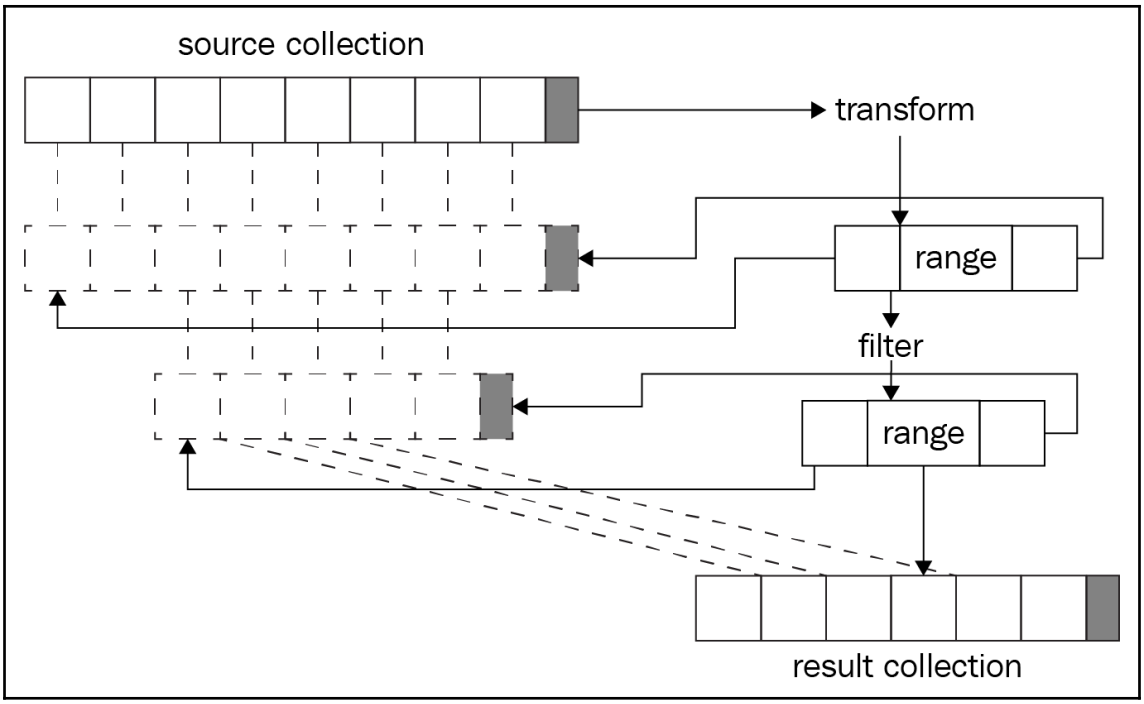
h	e	l	l	o	,		C	+	+	\0
0	1	2	3	4	5	6	7	8	9	10

Chapter 7: Functional Programming

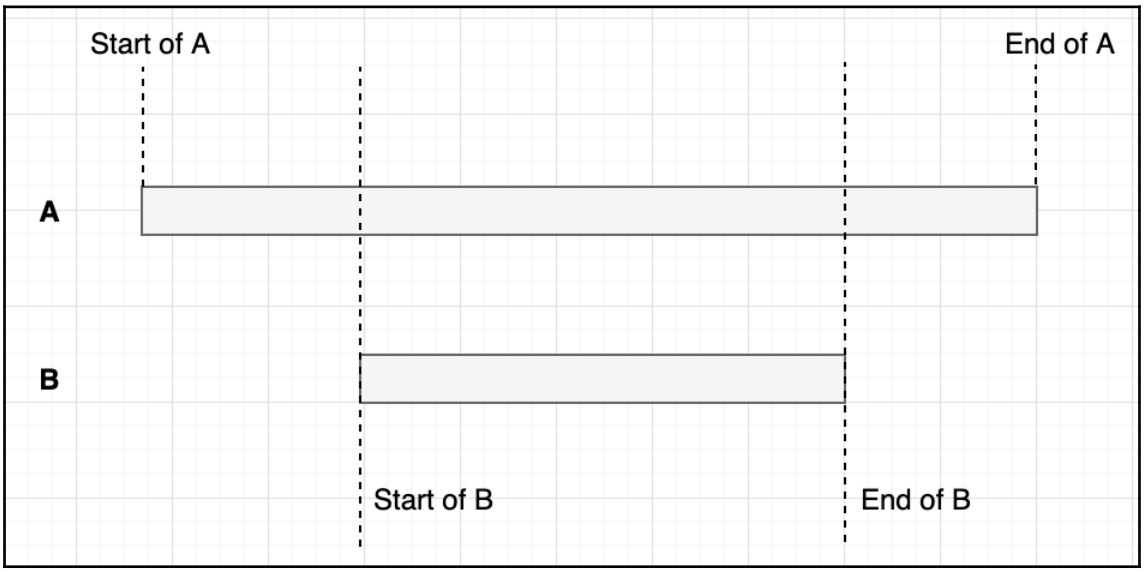


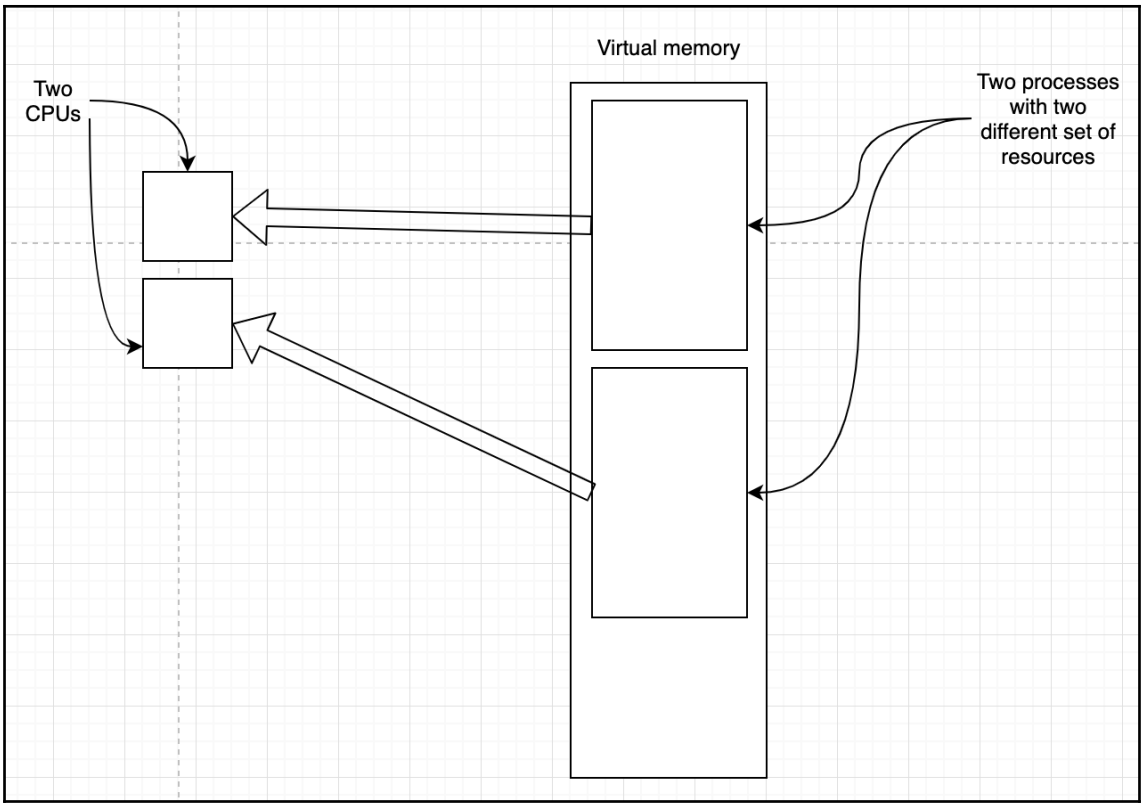


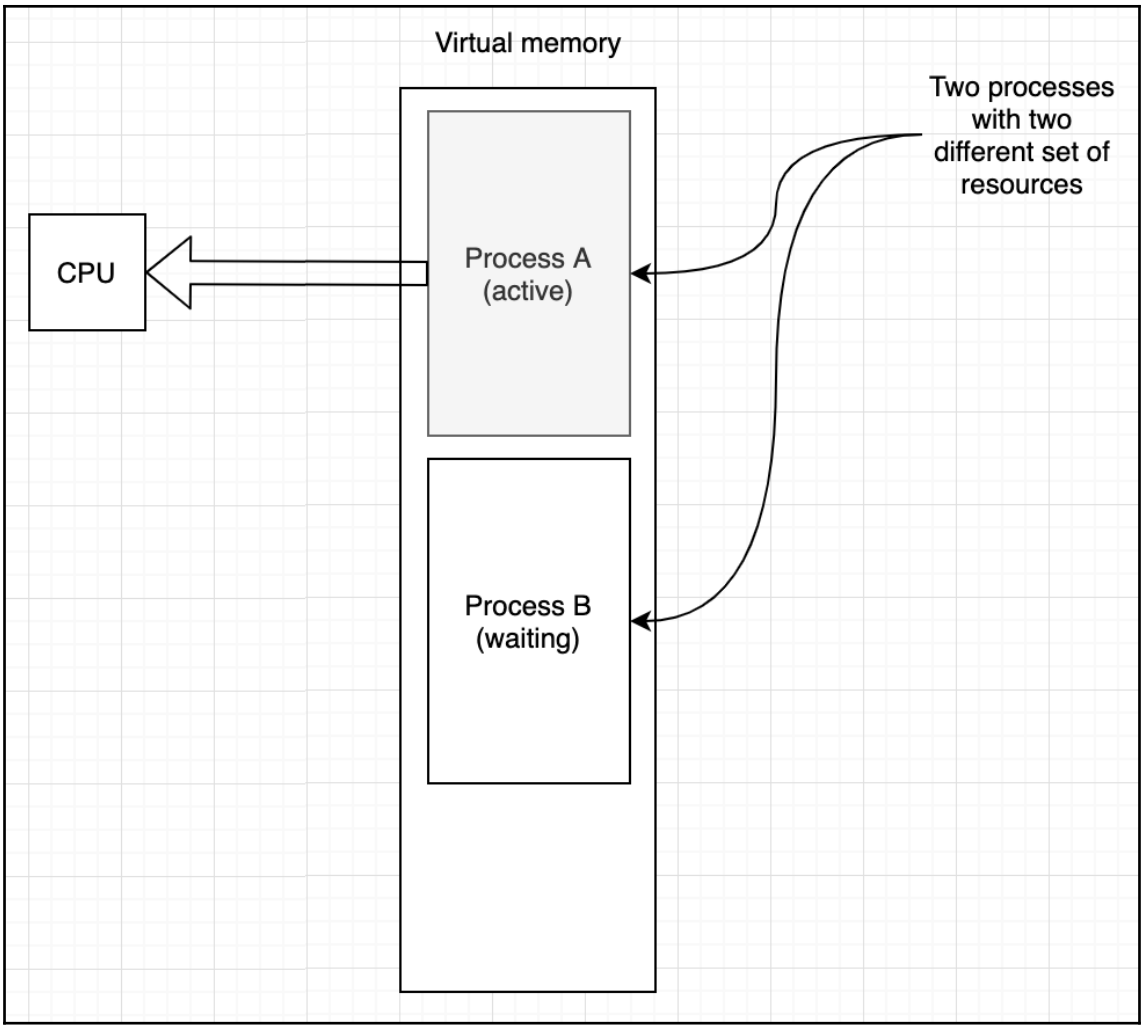


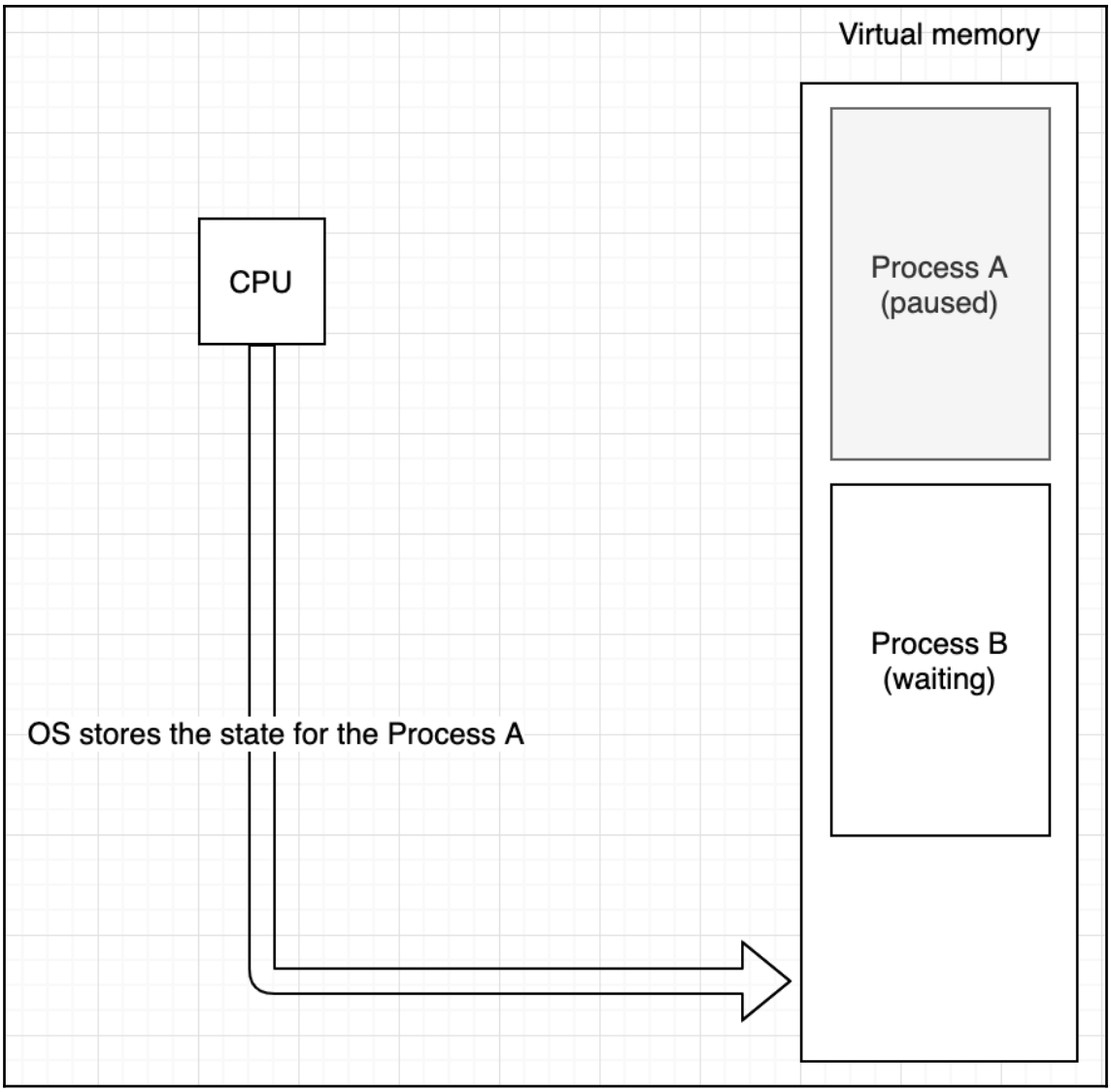

source collection
transform
range
filter
range
result collection

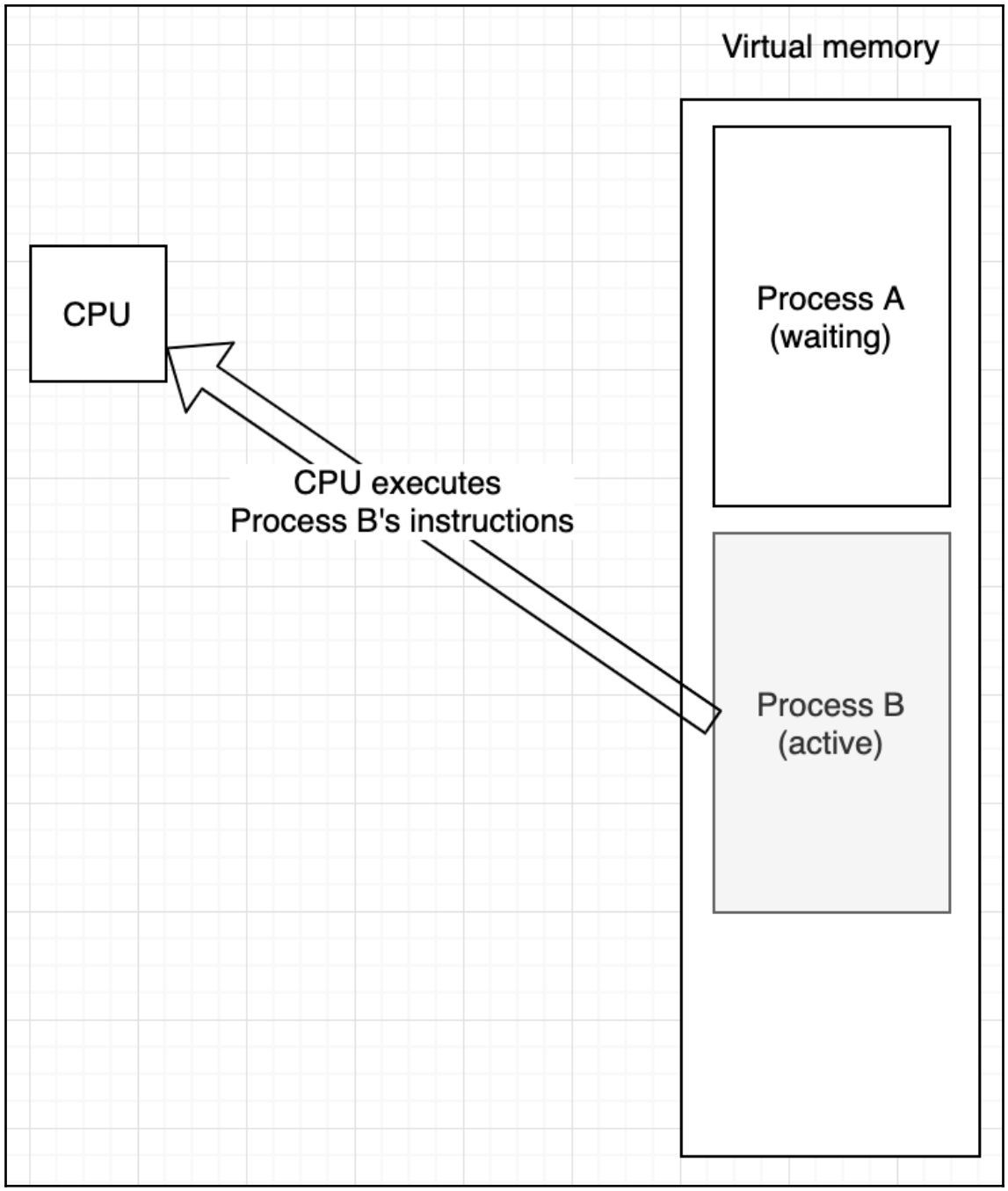
Chapter 8: Concurrency and Multithreading

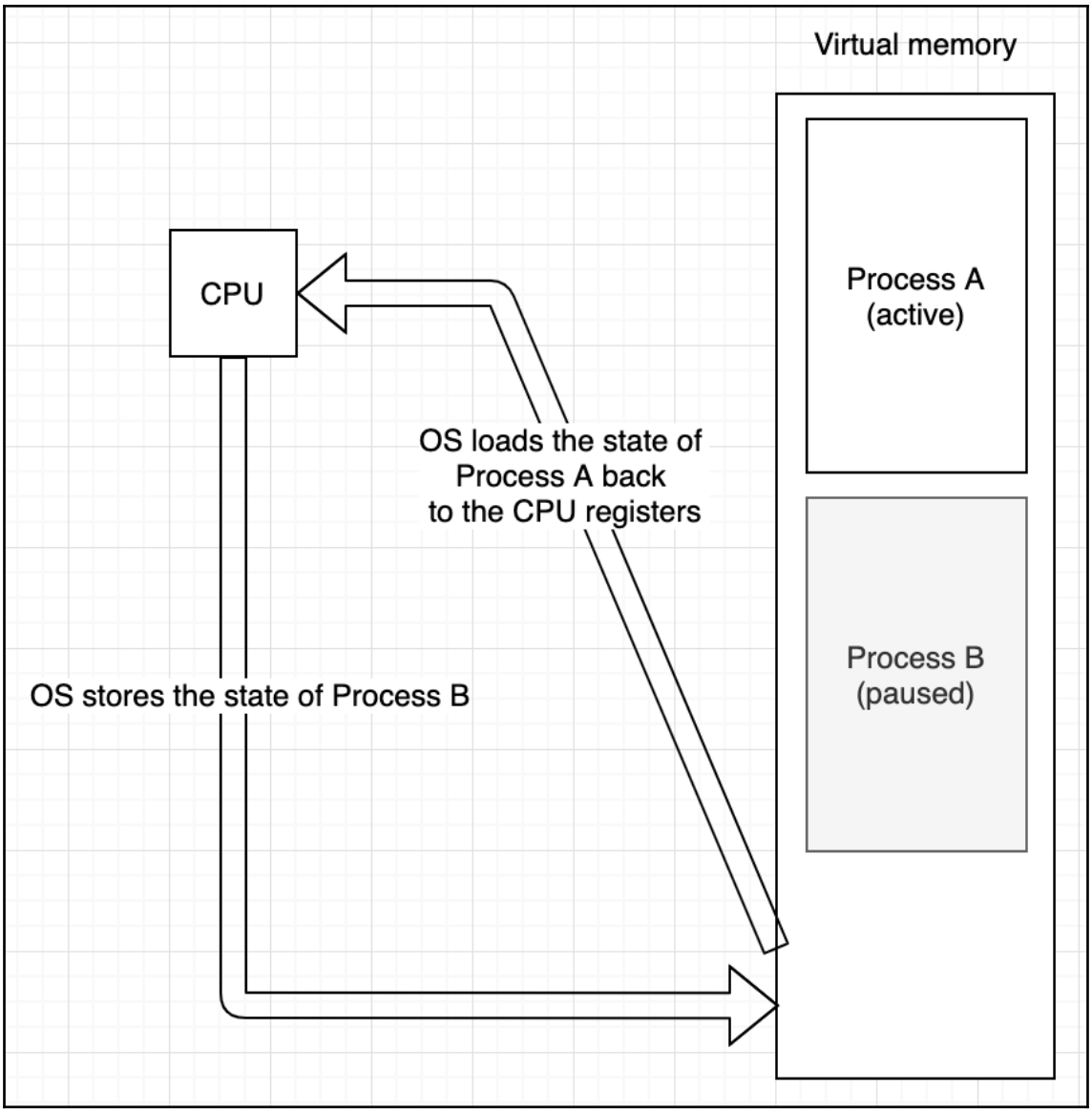


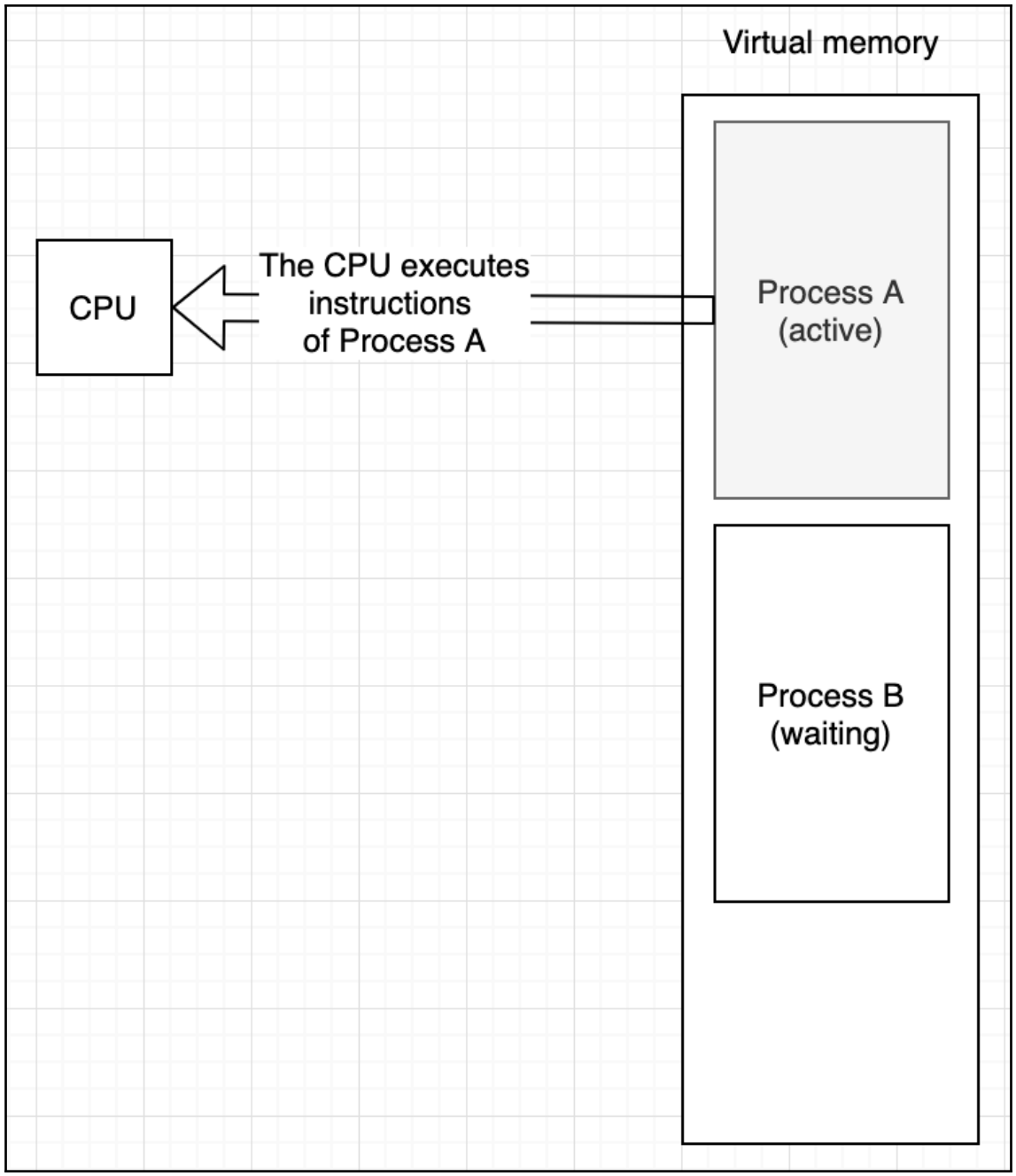

Virtual memory
Two
CPUs
Two processes
with two
different set of
resources

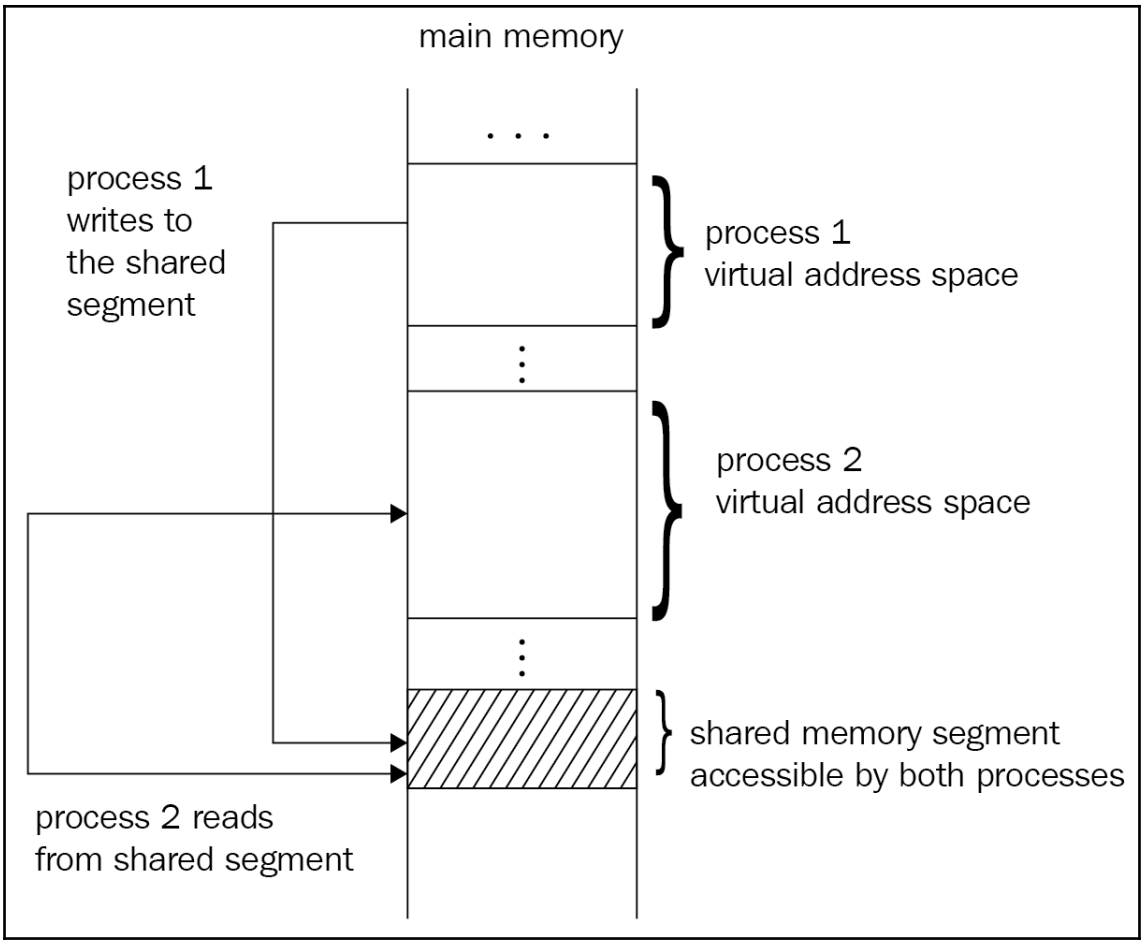

Virtual memory
Two processes with two different set of resources
CPU
Process A (active)
Process B (waiting)

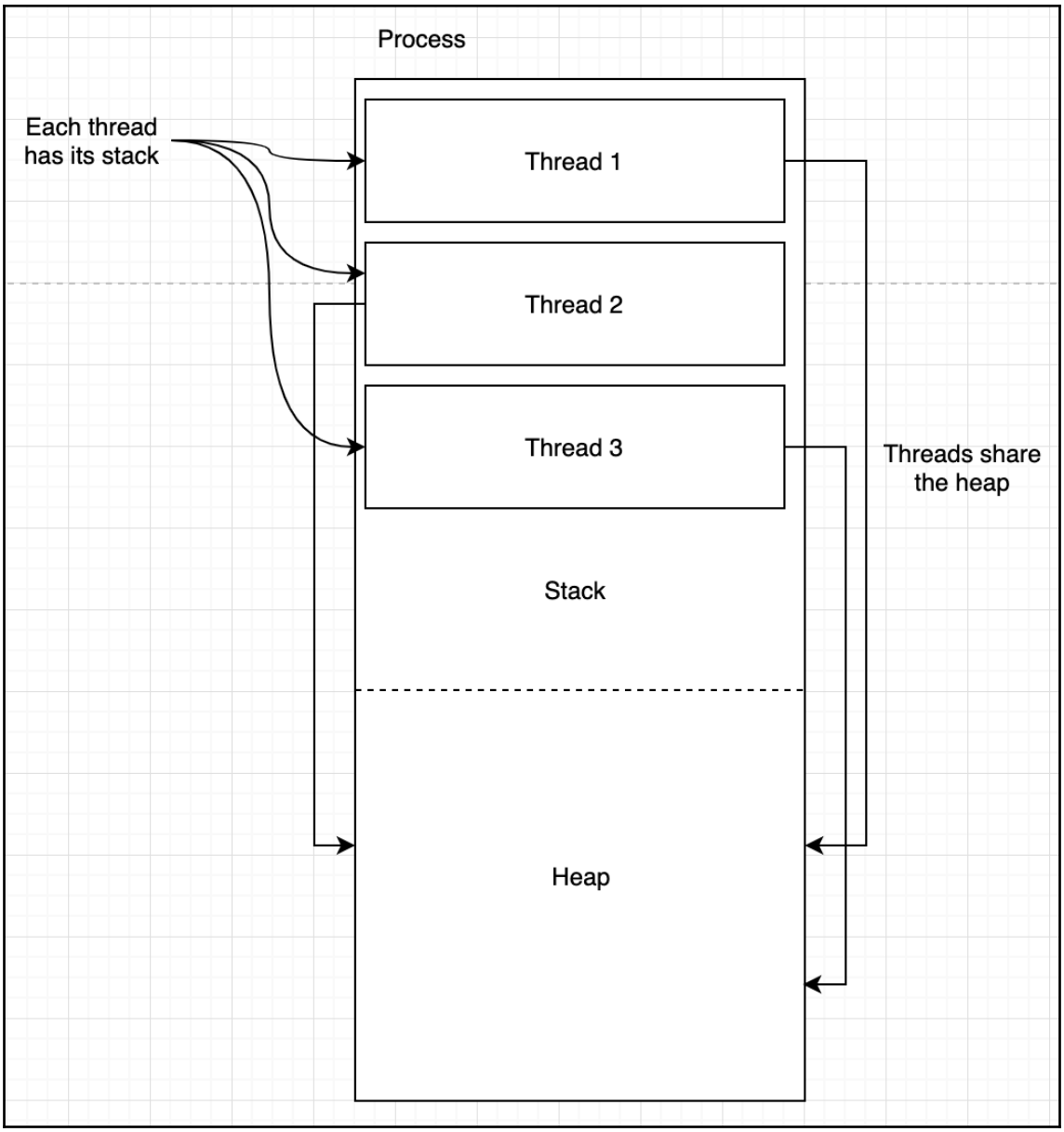

Virtual memory
CPU
Process A
(paused)
Process B
(waiting)
OS stores the state for the Process A

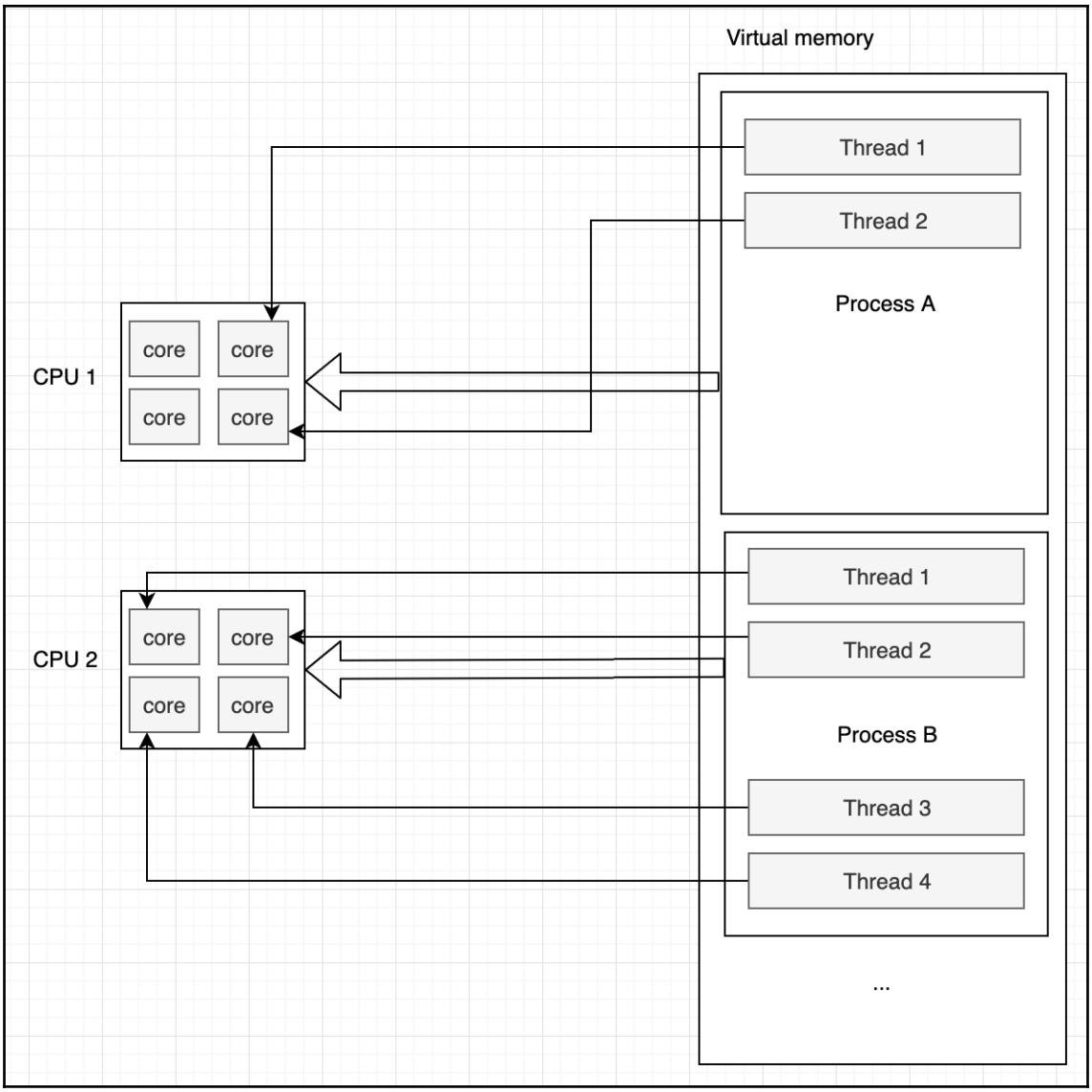

Virtual memory
CPU
Process A
(waiting)
CPU executes
Process B's instructions
Process B
(active)

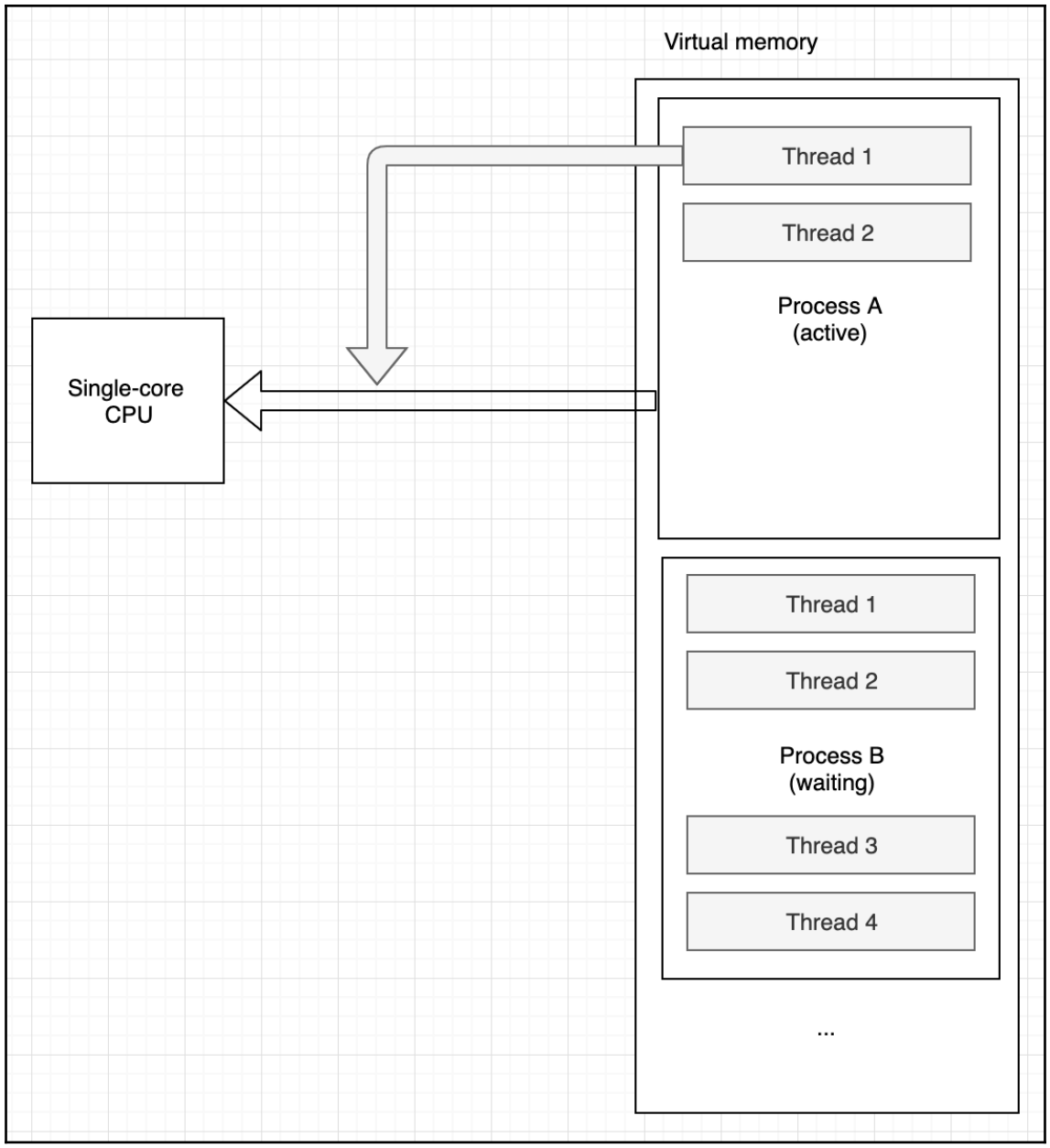

Virtual memory
CPU
Process A
(active)
OS loads the state of
Process A back
to the CPU registers
Process B
(paused)
OS stores the state of Process B

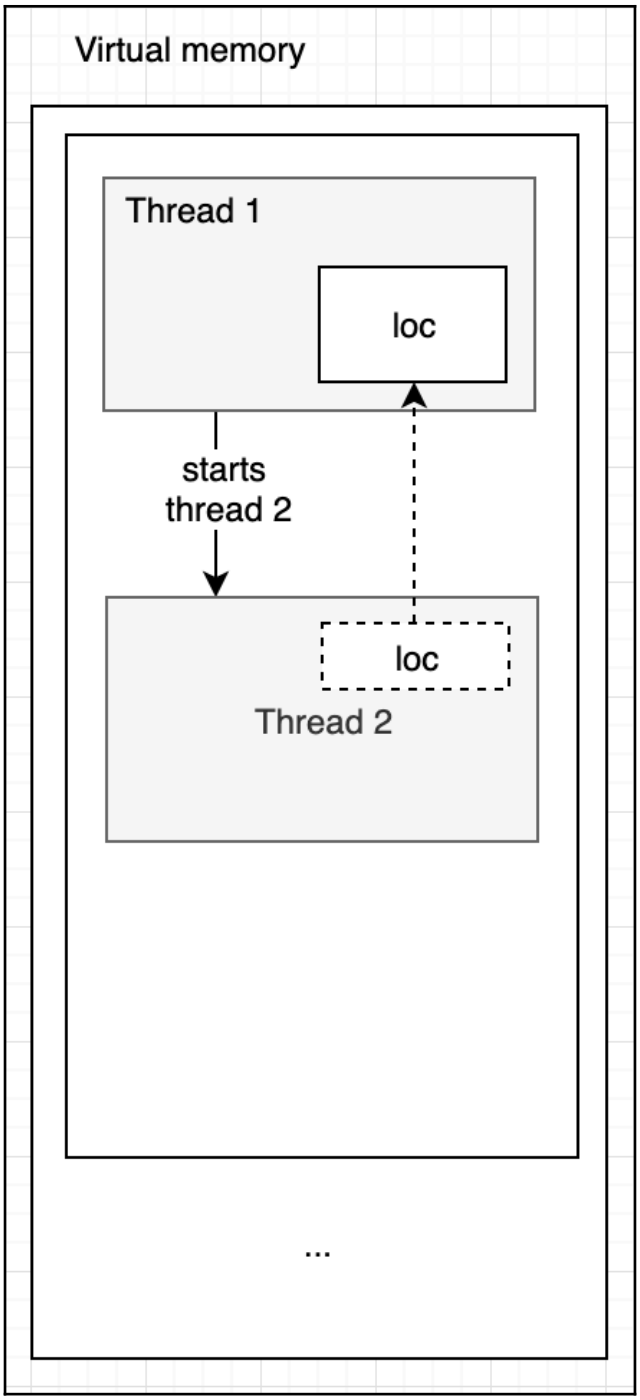

Virtual memory
CPU
The CPU executes
instructions
of Process A
Process A
(active)
Process B
(waiting)

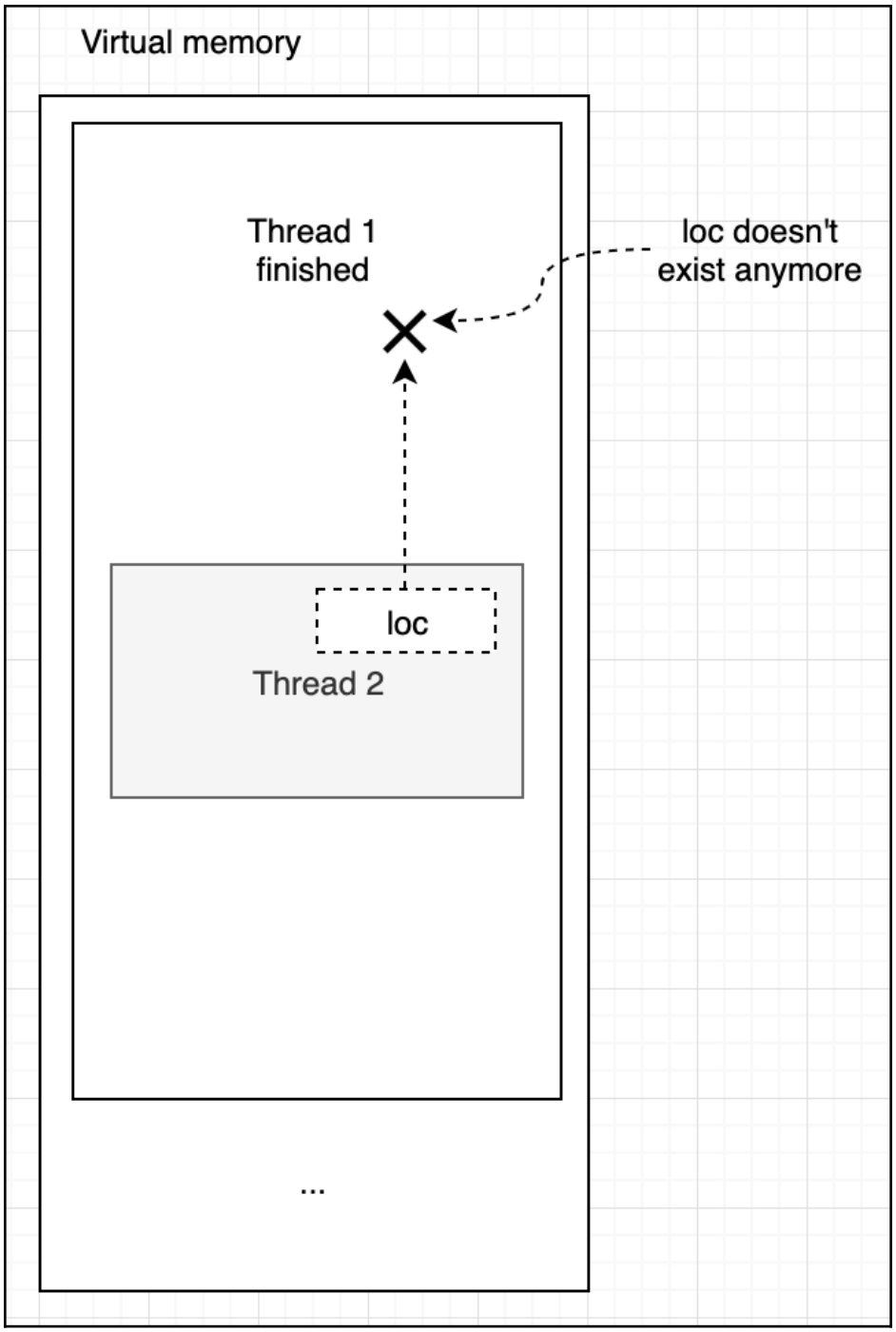

main memory
process 1
writes to
the shared
segment
process 1
virtual address space
process 2
virtual address space
shared memory segment
accessible by both processes
process 2 reads
from shared segment


Process
Each thread
has its stack
Thread 1
Thread 2
Thread 3
Threads share
the heap
Stack
Heap


Virtual memory
Thread 1
Thread 2
Process A
CPU 1
core
core
core
core
CPU 2
core
core
core
core
Thread 1
Thread 2
Process B
Thread 3
Thread 4
...


Virtual memory
Thread 1
Thread 2
Process A
(active)
Single-core
CPU
Thread 1
Thread 2
Process B
(waiting)
Thread 3
Thread 4
...

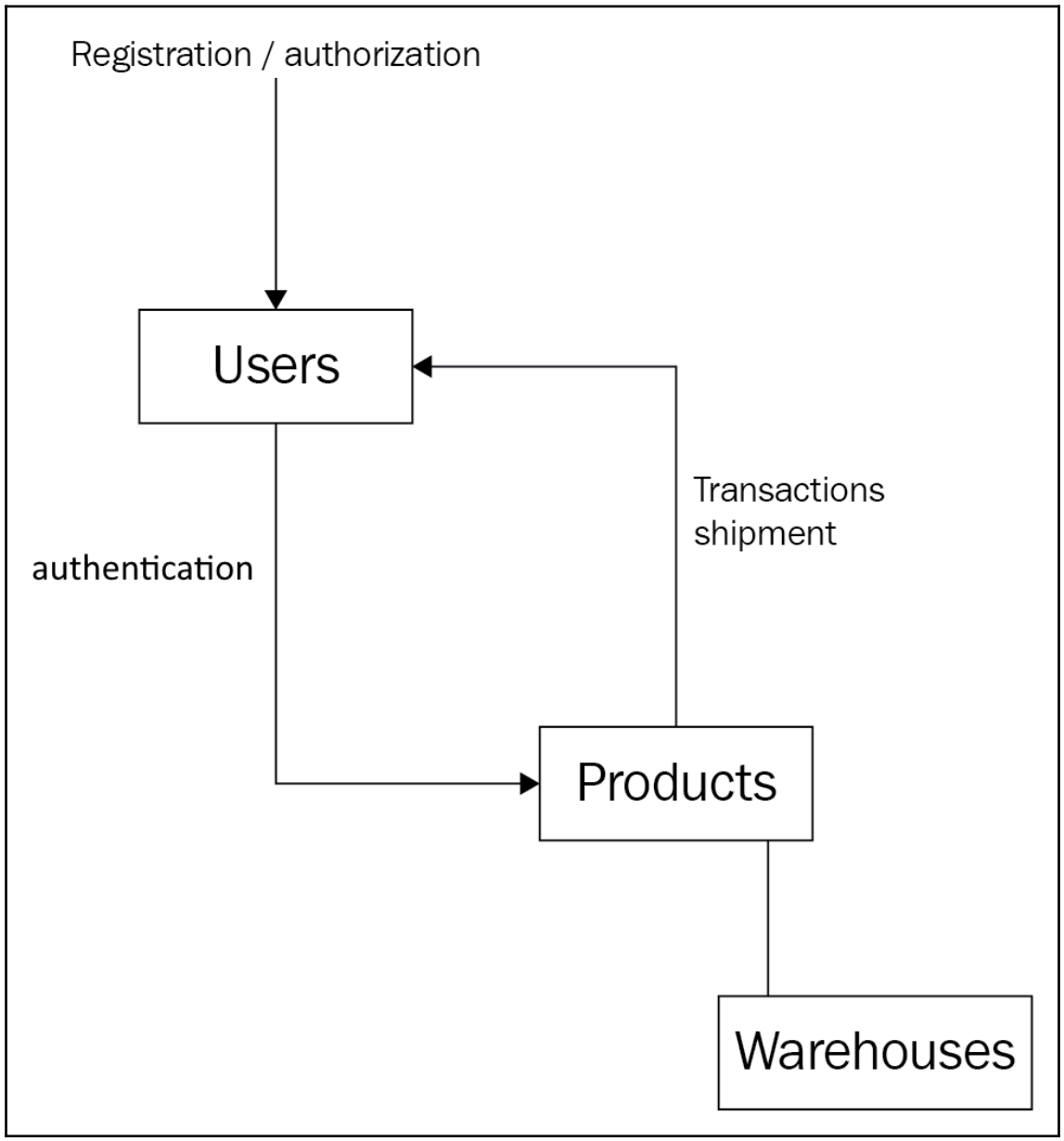

Virtual memory
Thread 1
loc
starts
thread 2
loc
Thread 2
...

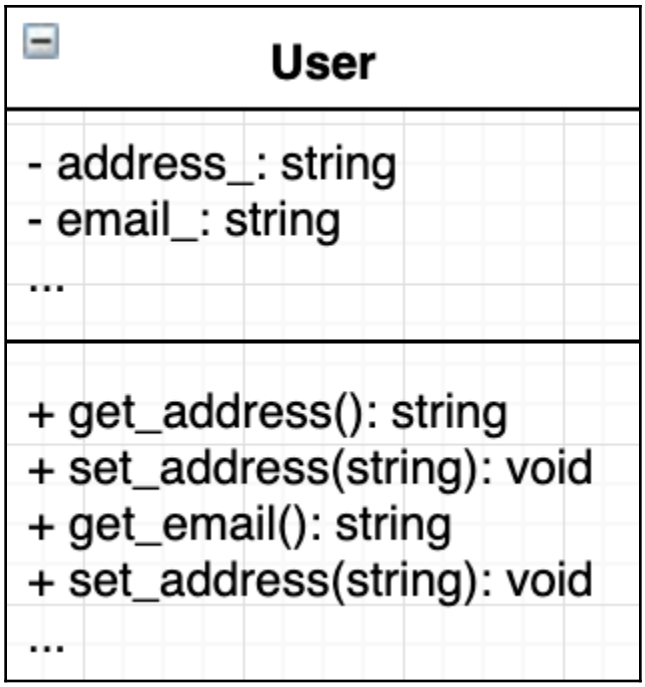

Virtual memory
Thread 1
finished
loc doesn't
exist anymore
loc
Thread 2
...

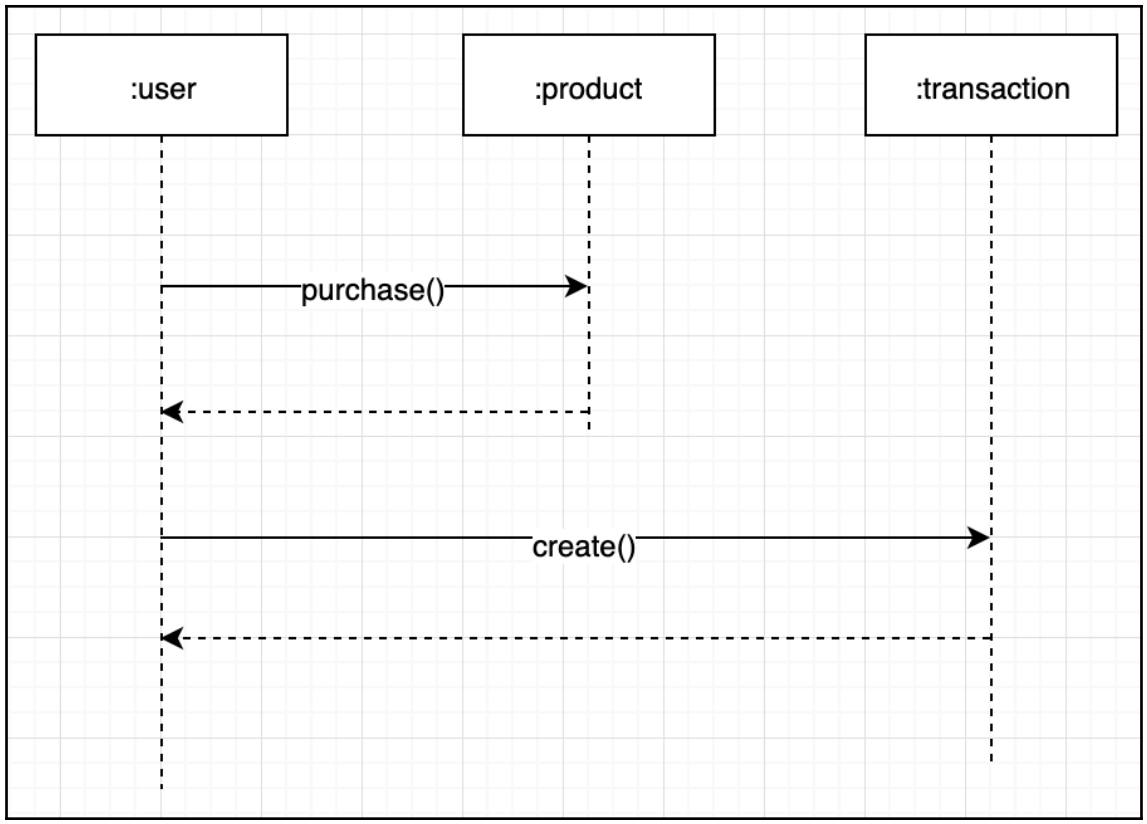
Chapter 9: Designing Concurrent Data Structures

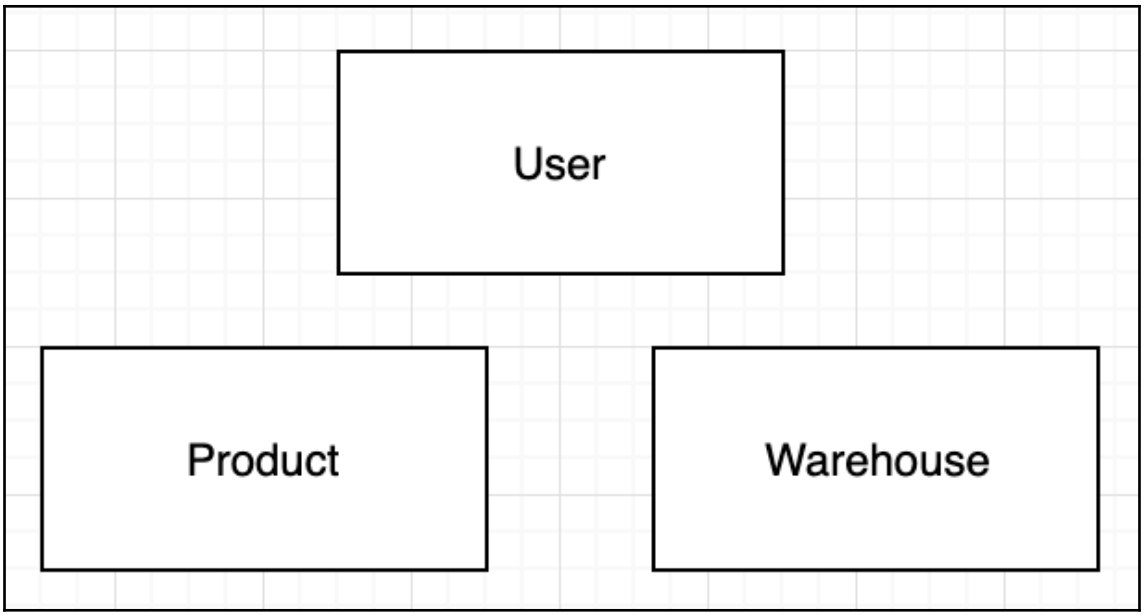
No images...

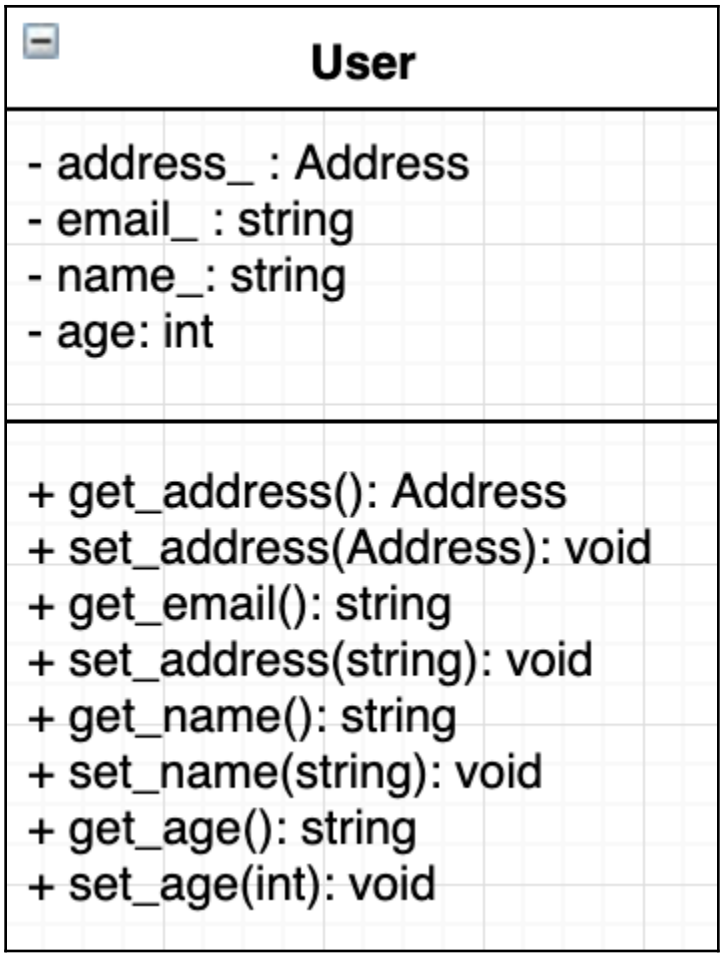
Chapter 10: Designing World-Ready Applications

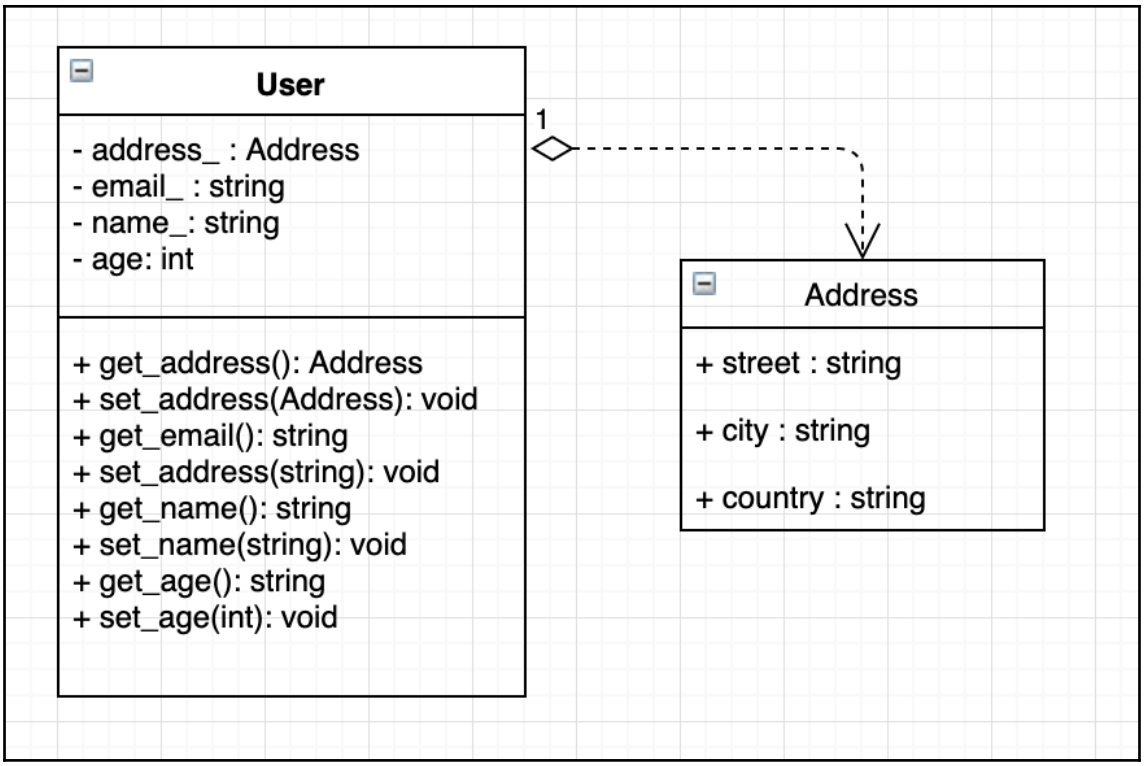


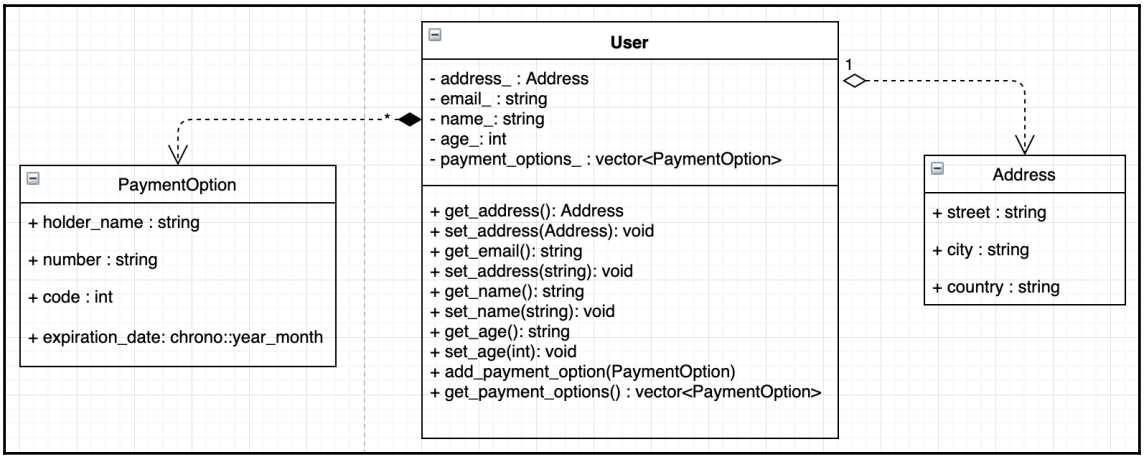

User
- address_: string
- email_: string
...
+ get_address(): string
+ set_address(string): void
+ get_email(): string
+ set_address(string): void
...

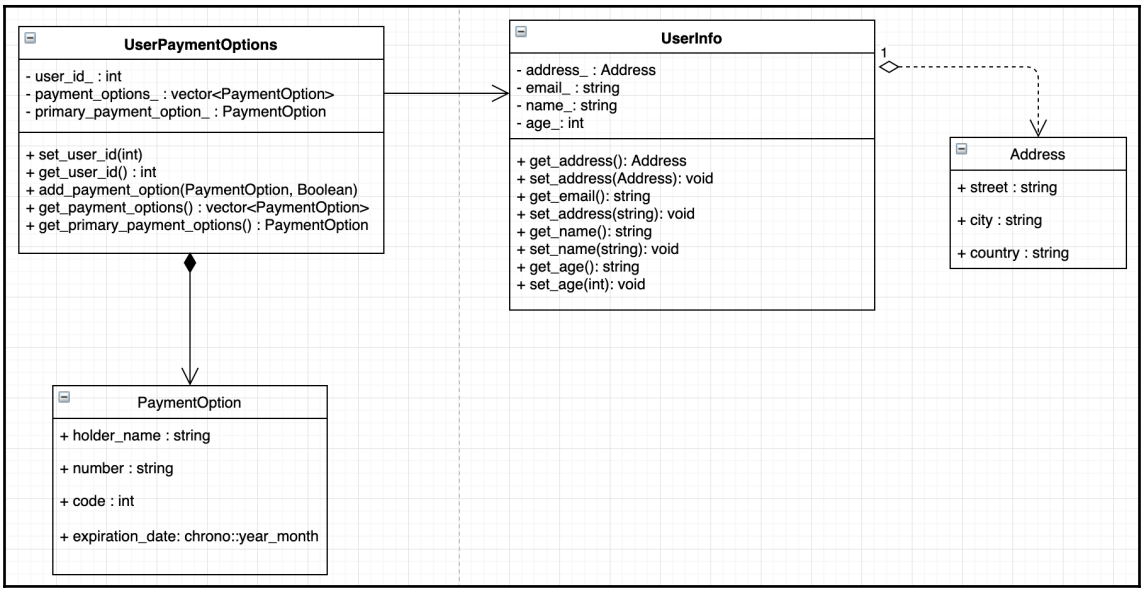

:user
:product
:transaction
purchase()
create()

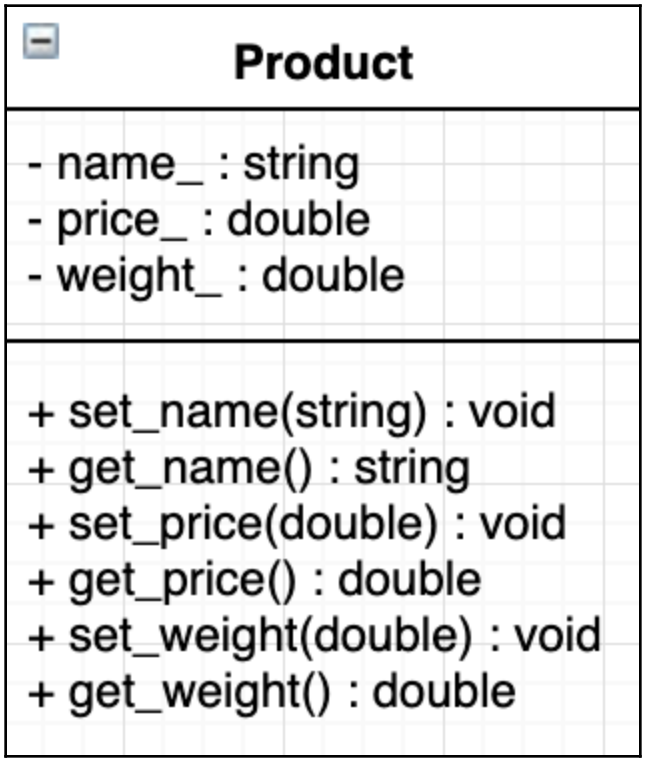

User
Product
Warehouse

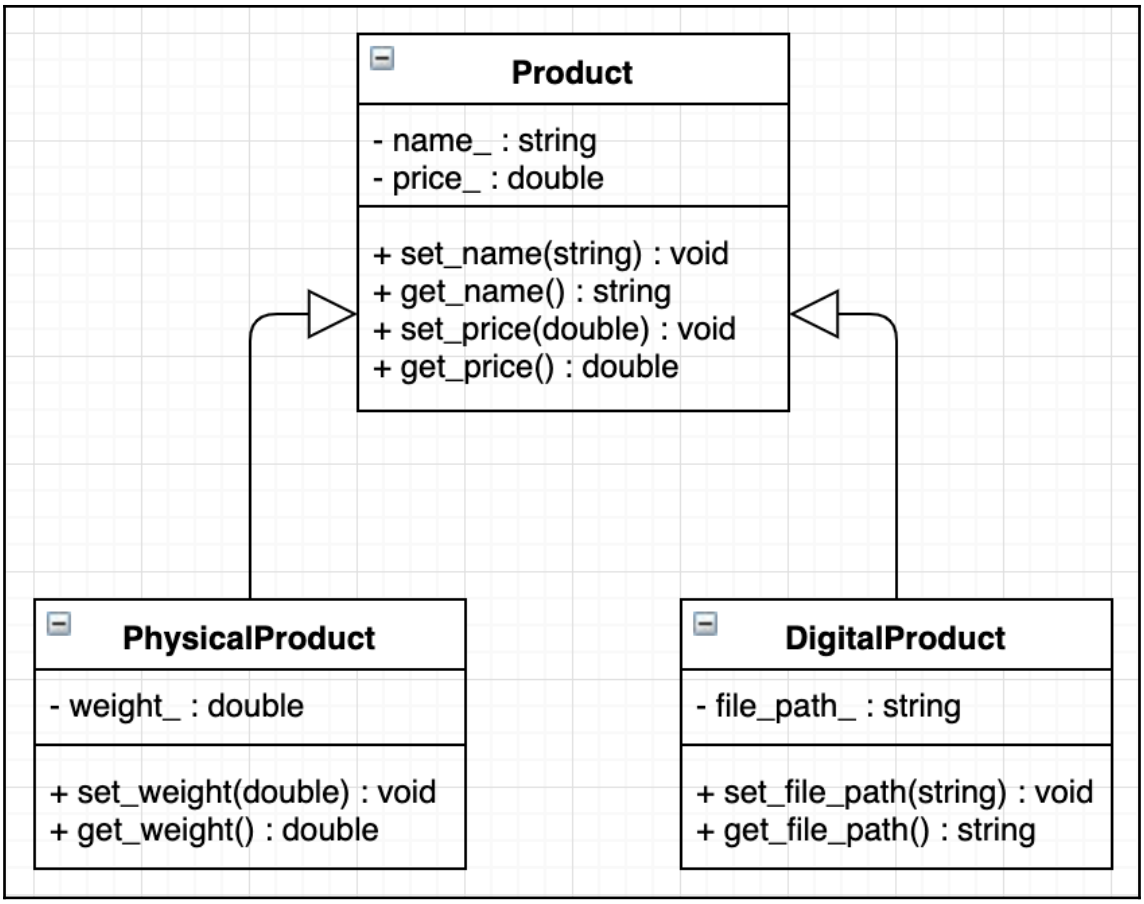

User
- address_ : Address
- email_ : string
- name_: string
- age: int
+ get_address(): Address
+ set_address(Address): void
+ get_email(): string
+ set_address(string): void
+ get_name(): string
+ set_name(string): void
+ get_age(): string
+ set_age(int): void

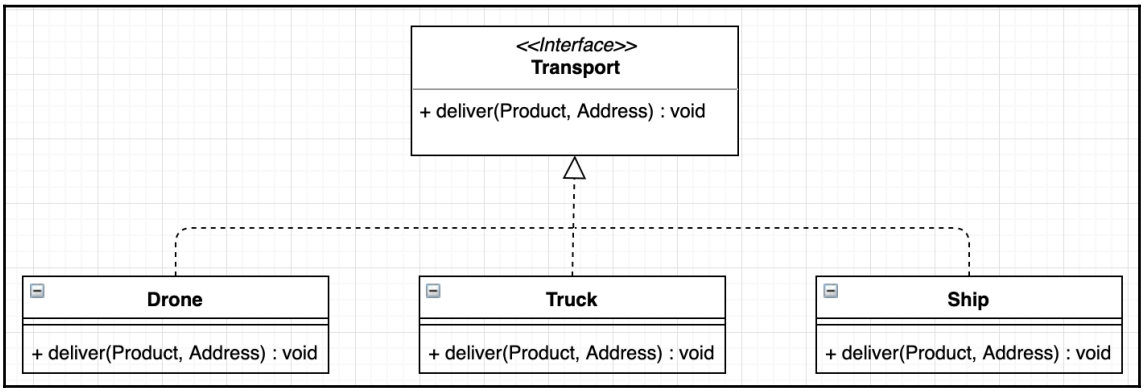

User
- address_ : Address
- email_ : string
- name_: string
- age: int
+ get_address(): Address
+ set_address(Address): void
+ get_email(): string
+ set_address(string): void
+ get_name(): string
+ set_name(string): void
+ get_age(): string
+ set_age(int): void
1
Address
+ street : string
+ city : string
+ country : string

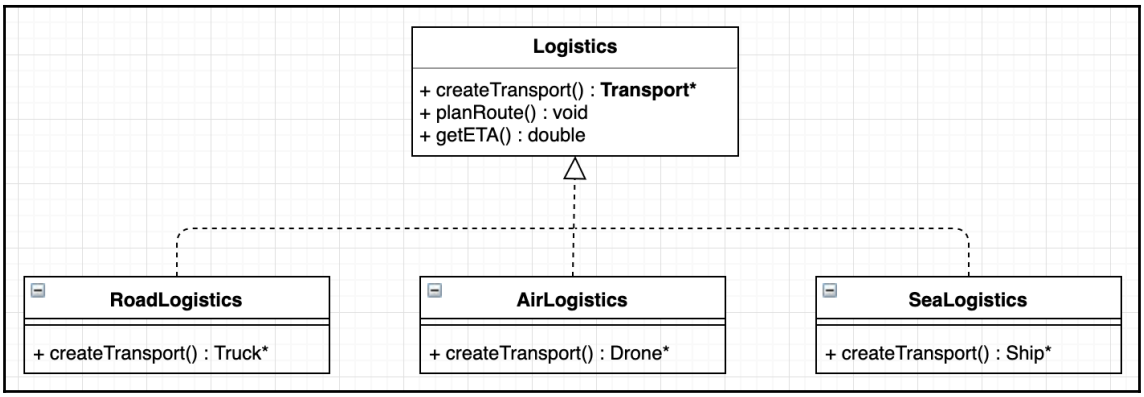

User
- address_ : Address
- email_ : string
- name_: string
- age_: int
- payment_options_ : vector<PaymentOption>
+ get_address(): Address
+ set_address(Address): void
+ get_email(): string
+ set_address(string): void
+ get_name(): string
+ set_name(string): void
+ get_age(): string
+ set_age(int): void
+ add_payment_option(PaymentOption)
+ get_payment_options() : vector<PaymentOption>
1
*
PaymentOption
+ holder_name : string
+ number : string
+ code : int
+ expiration_date: chrono::year_month
Address
+ street : string
+ city : string
+ country : string


UserPaymentOptions
- user_id_ : int
- payment_options_ : vector<PaymentOption>
- primary_payment_option_ : PaymentOption
+ set_user_id(int)
+ get_user_id() : int
+ add_payment_option(PaymentOption, Boolean)
+ get_payment_options() : vector<PaymentOption>
+ get_primary_payment_options() : PaymentOption
UserInfo
- address_ : Address
- email_ : string
- name_: string
- age_: int
+ get_address(): Address
+ set_address(Address): void
+ get_email(): string
+ set_address(string): void
+ get_name(): string
+ set_name(string): void
+ get_age(): string
+ set_age(int): void
1
Address
+ street : string
+ city : string
+ country : string
PaymentOption
+ holder_name : string
+ number : string
+ code : int
+ expiration_date: chrono::year_month

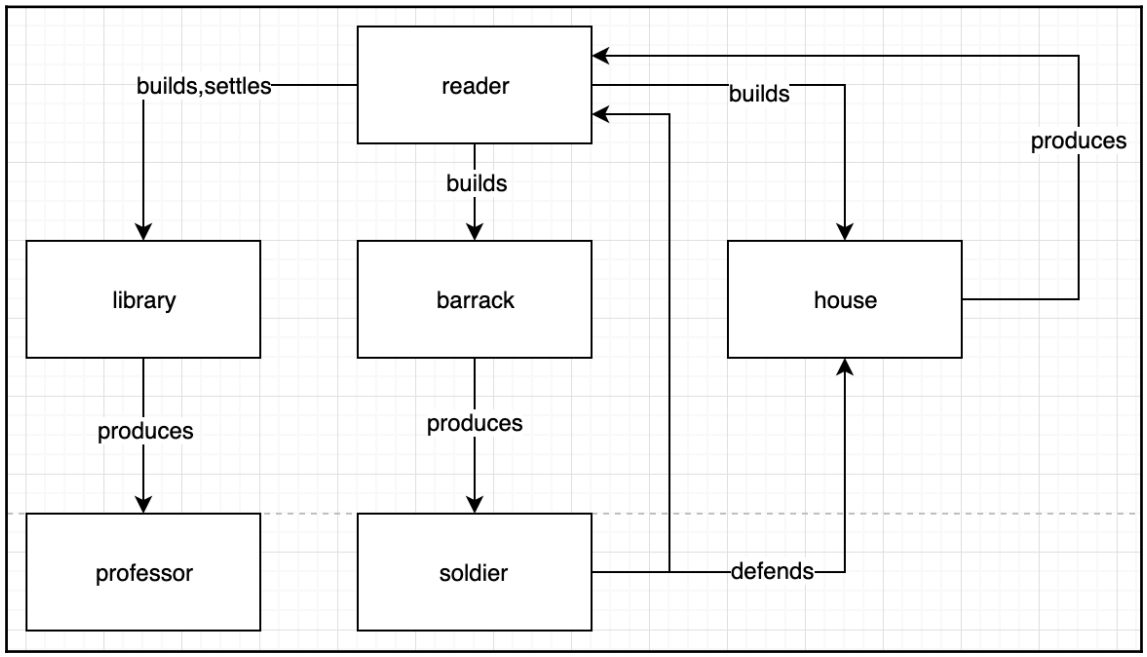

Product
- name_ : string
- price_ : double
- weight_ : double
+ set_name(string) : void
+ get_name() : string
+ set_price(double) : void
+ get_price() : double
+ set_weight(double) : void
+ get_weight() : double

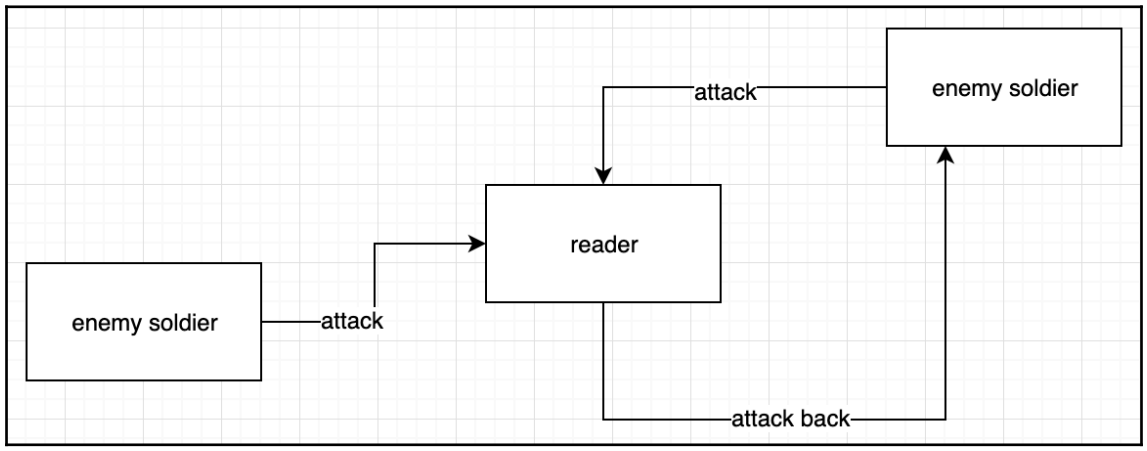

Product
- name_ : string
- price_ : double
+ set_name(string) : void
+ get_name() : string
+ set_price(double) : void
+ get_price() : double
PhysicalProduct
- weight_ : double
+ set_weight(double) : void
+ get_weight() : double
DigitalProduct
- file_path_ : string
+ set_file_path(string) : void
+ get_file_path() : string

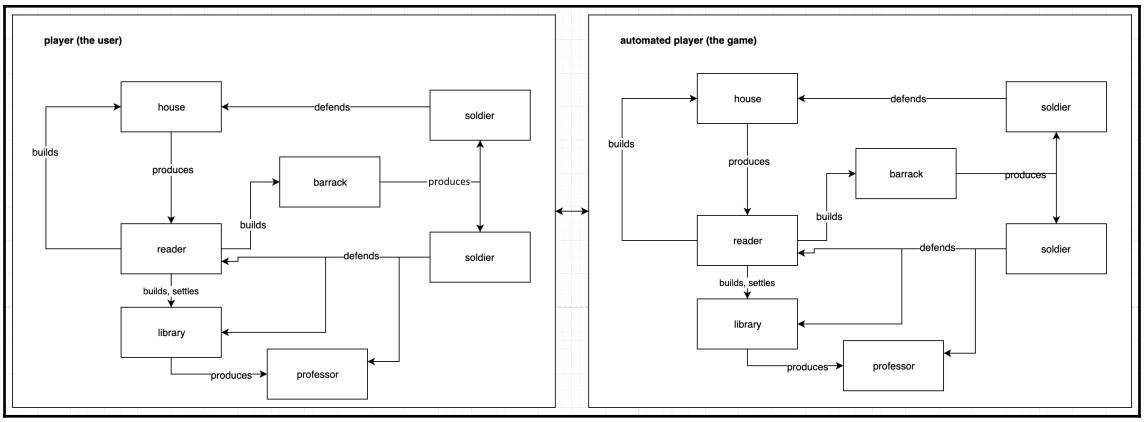

<<Interface>>
Transport
+ deliver(Product, Address) : void
Drone
+ deliver(Product, Address) : void
Truck
+ deliver(Product, Address) : void
Ship
+ deliver(Product, Address) : void

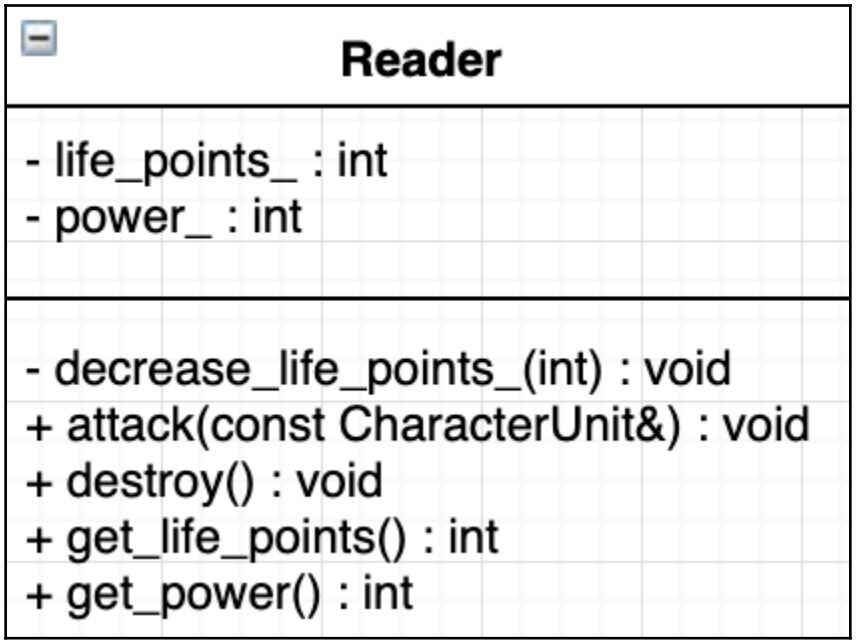

Logistics
+ createTransport() : Transport*
+ planRoute() : void
+ getETA() : double
RoadLogistics
+ createTransport() : Truck*
AirLogistics
+ createTransport() : Drone*
SeaLogistics
+ createTransport() : Ship*

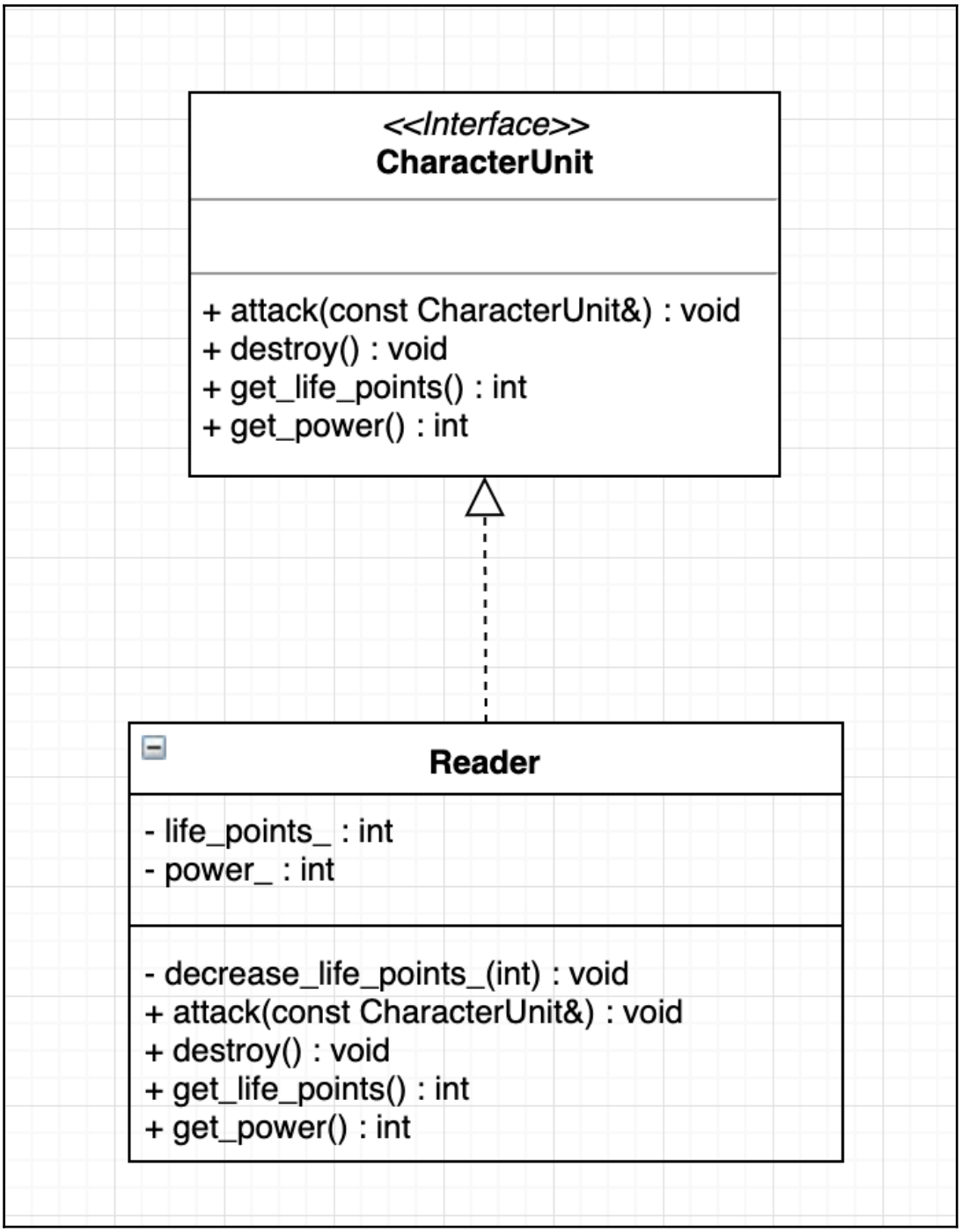
Chapter 11: Designing a Strategy Game Using Design Patterns

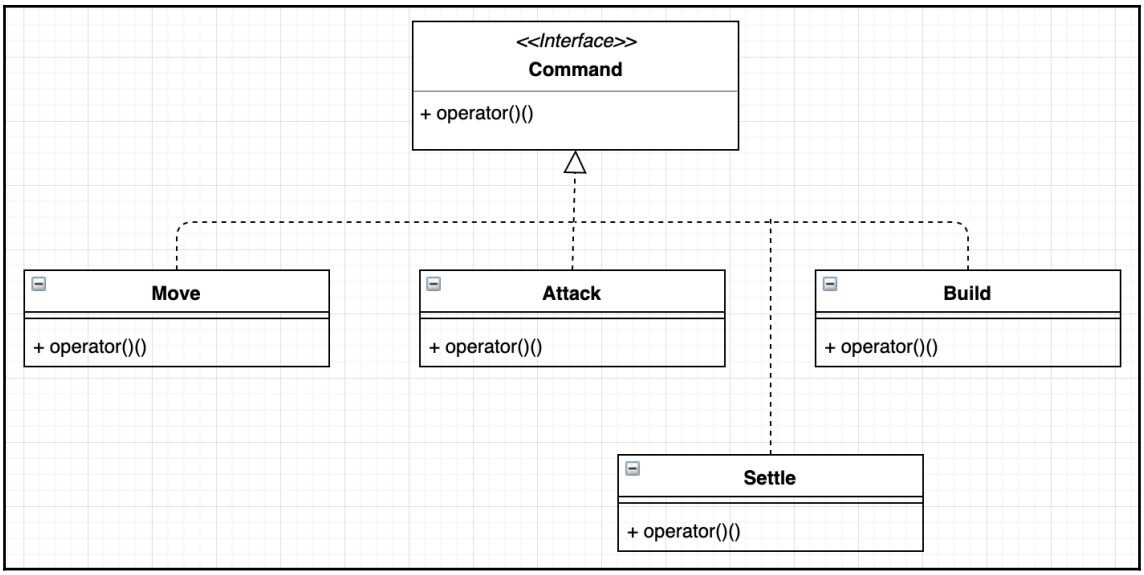


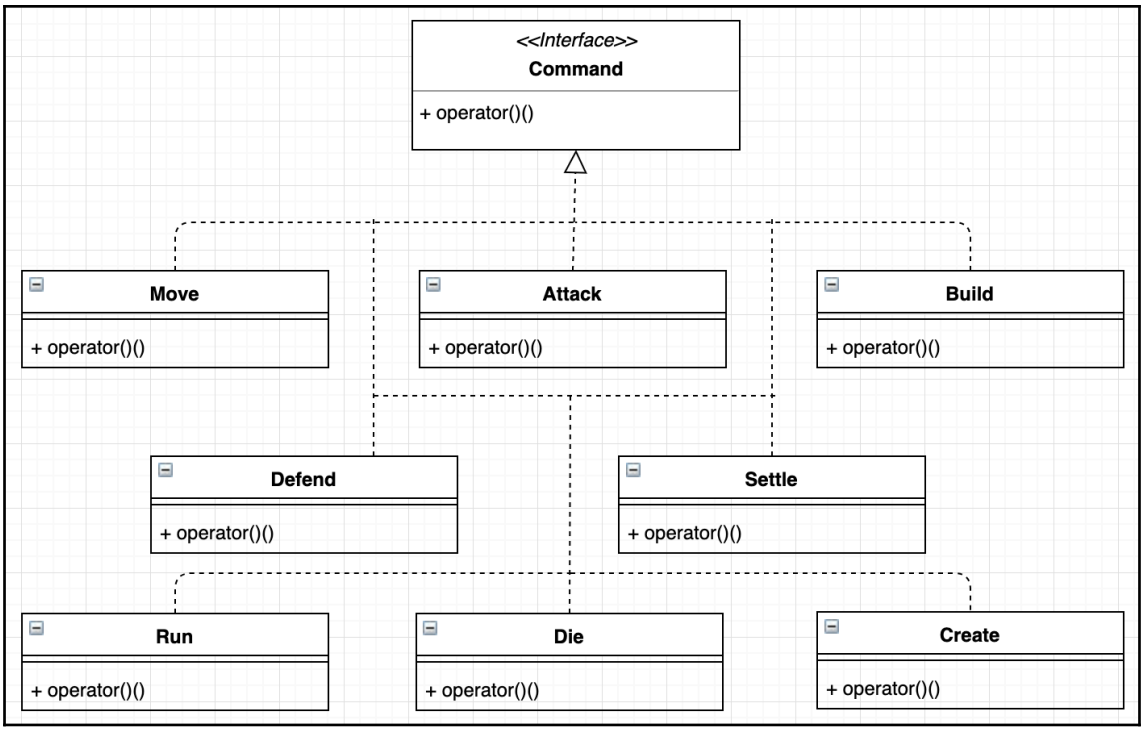


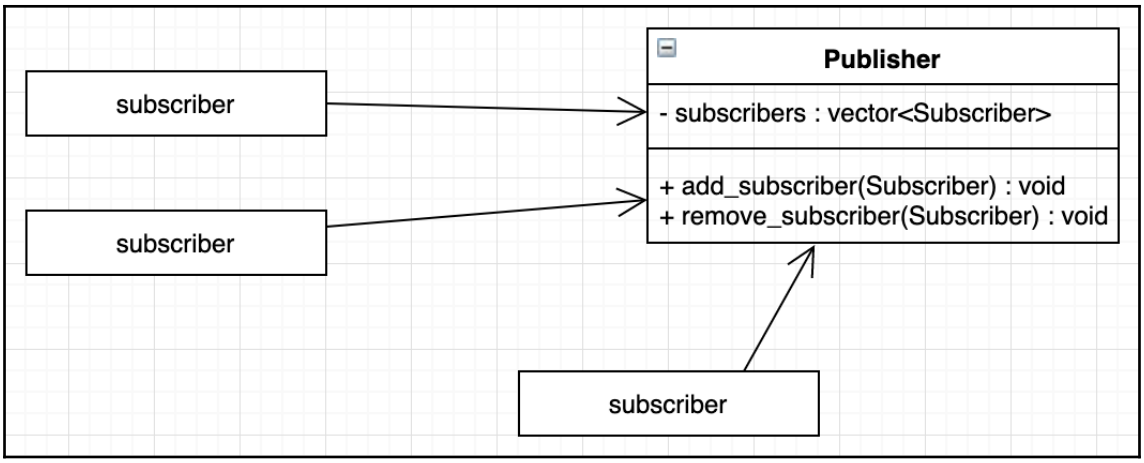

player (the user)
house
defends
soldier
builds
produces
barrack
produces
builds
reader
defends
soldier
builds, settles
library
produces
professor
automated player (the game)
house
defends
soldier
builds
produces
barrack
produces
builds
reader
defends
soldier
builds, settles
library
produces
professor

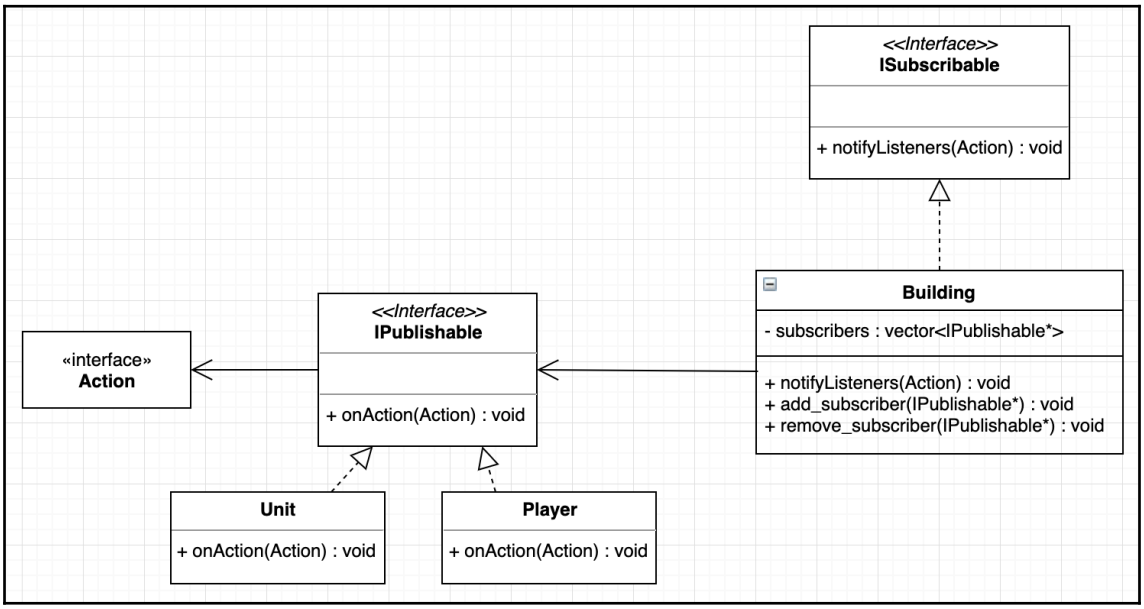

Reader
- life_points_ : int
- power_ : int
- decrease_life_points_(int) : void
+ attack(const CharacterUnit&) : void
+ destroy() : void
+ get_life_points() : int
+ get_power() : int

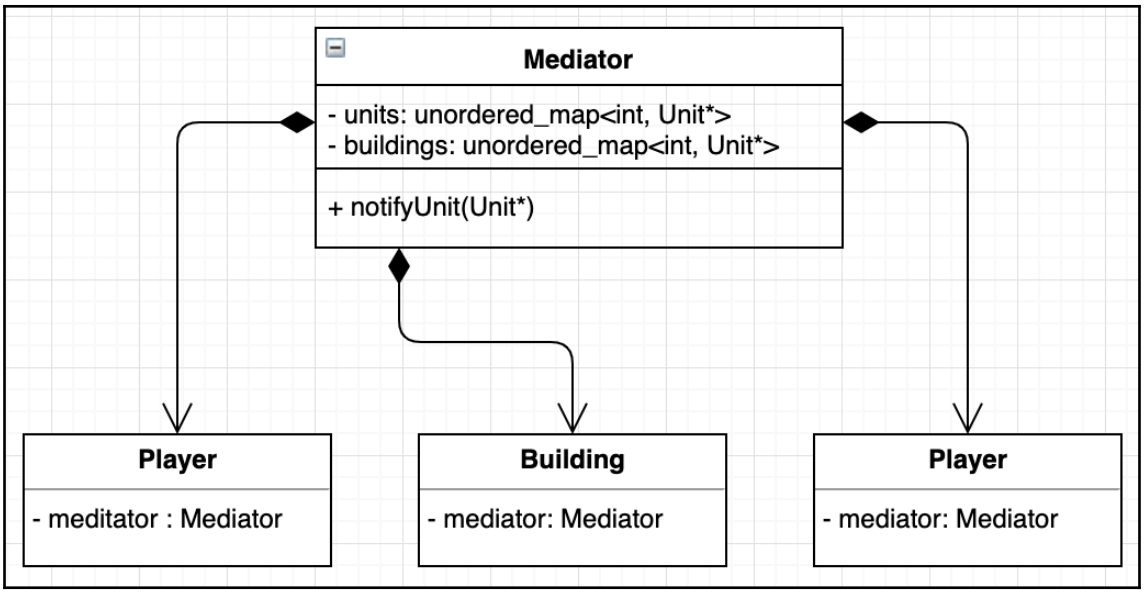

<<Interface>>
CharacterUnit
+ attack(const CharacterUnit&) : void
+ destroy() : void
+ get_life_points() : int
+ get_power() : int
Reader
- life_points_ : int
- power_ : int
- decrease_life_points_(int) : void
+ attack(const CharacterUnit&) : void
+ destroy() : void
+ get_life_points() : int
+ get_power() : int

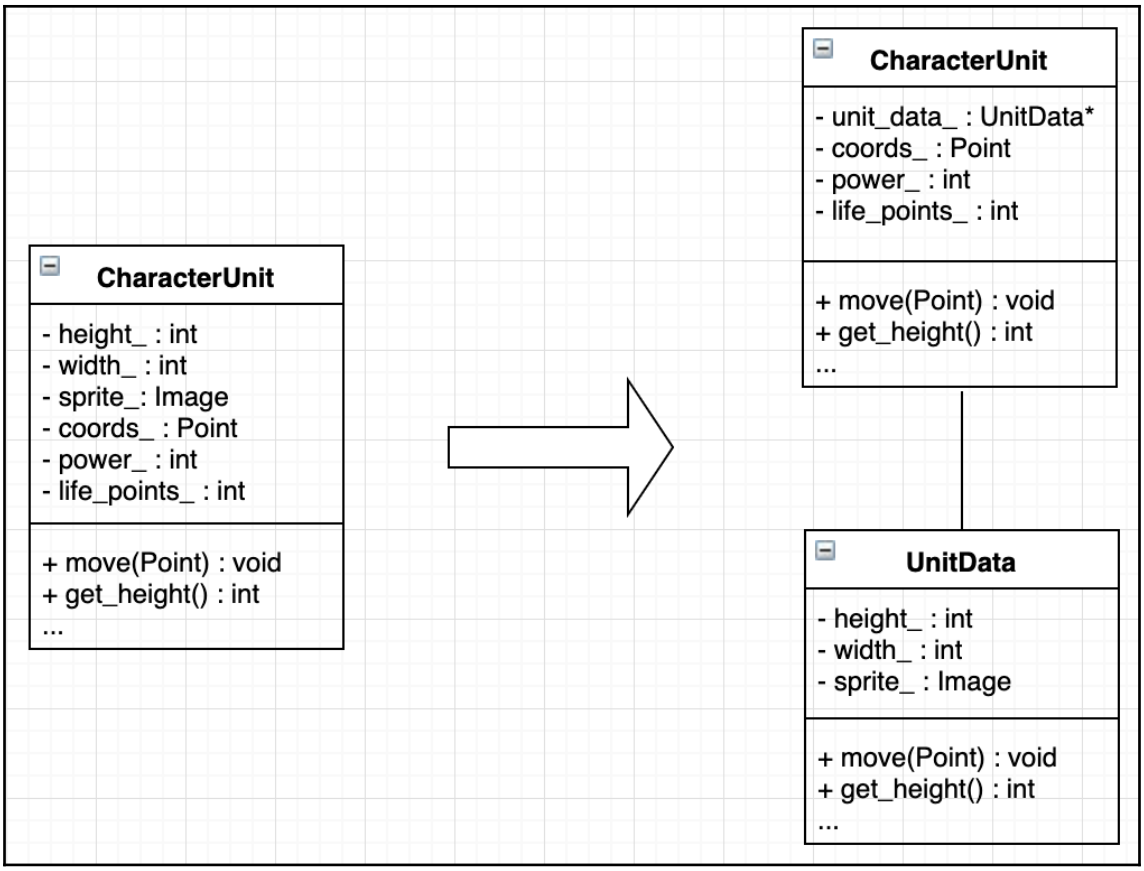

<<Interface>>
Command
+ operator()()
Move
+ operator()()
Attack
+ operator()()
Build
+ operator()()
Settle
+ operator()()

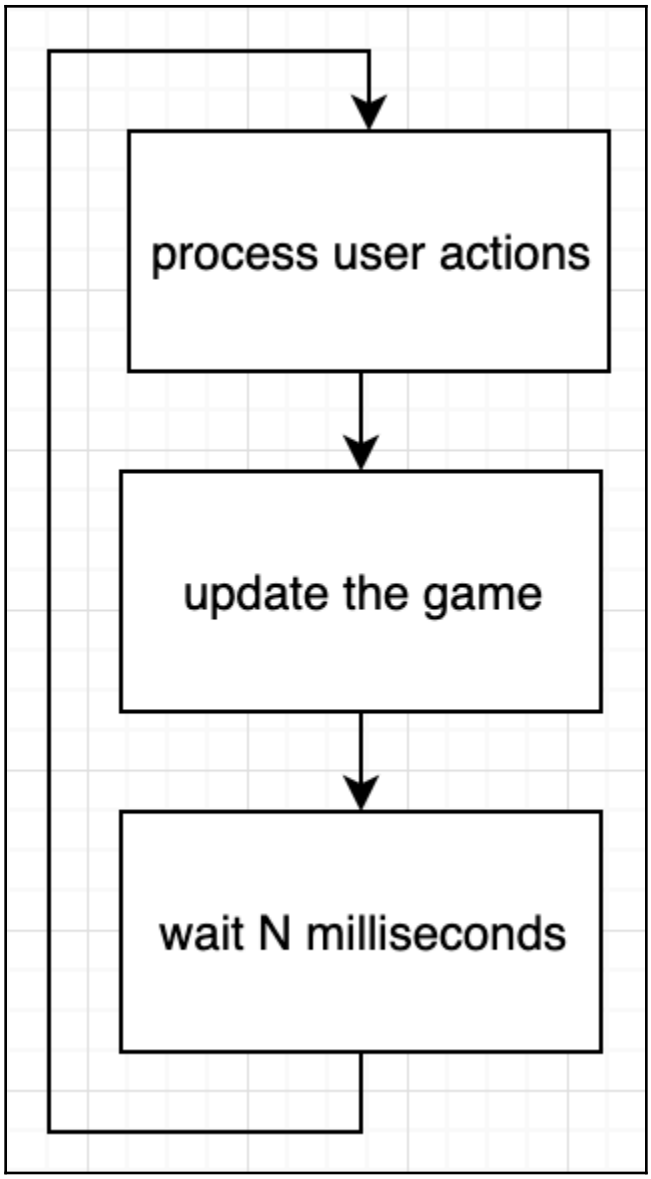

<<Interface>>
Command
+ operator()()
Move
+ operator()()
Attack
+ operator()()
Build
+ operator()()
Defend
+ operator()()
Settle
+ operator()()
Run
+ operator()()
Die
+ operator()()
Create
+ operator()()


Publisher
- subscribers : vector<Subscriber>
+ add_subscriber(Subscriber) : void
+ remove_subscriber(Subscriber) : void
subscriber
subscriber
subscriber

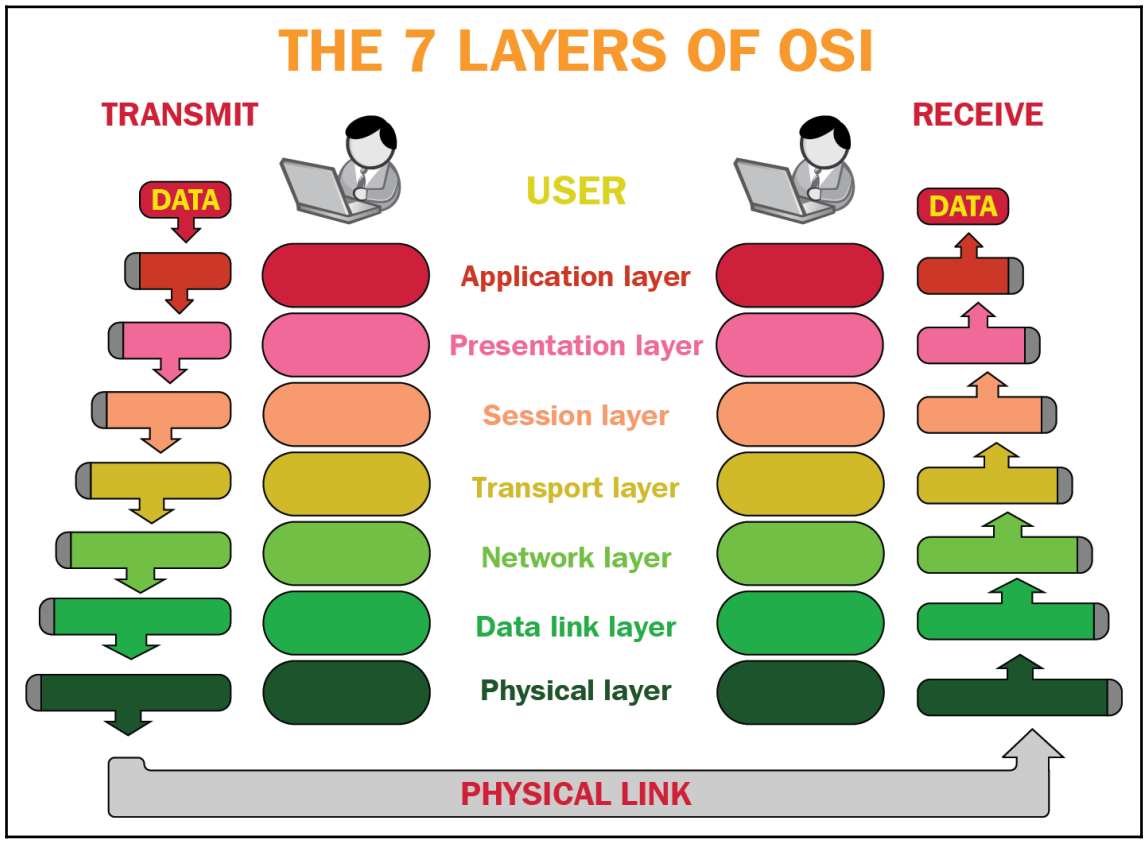

<<Interface>>
ISubscribable
+ notifyListeners(Action) : void
Building
- subscribers : vector<IPublishable*>
+ notifyListeners(Action) : void
+ add_subscriber(IPublishable*) : void
+ remove_subscriber(IPublishable*) : void
<<Interface>>
IPublishable
+ onAction(Action) : void
«interface»
Action
Unit
+ onAction(Action) : void
Player
+ onAction(Action) : void

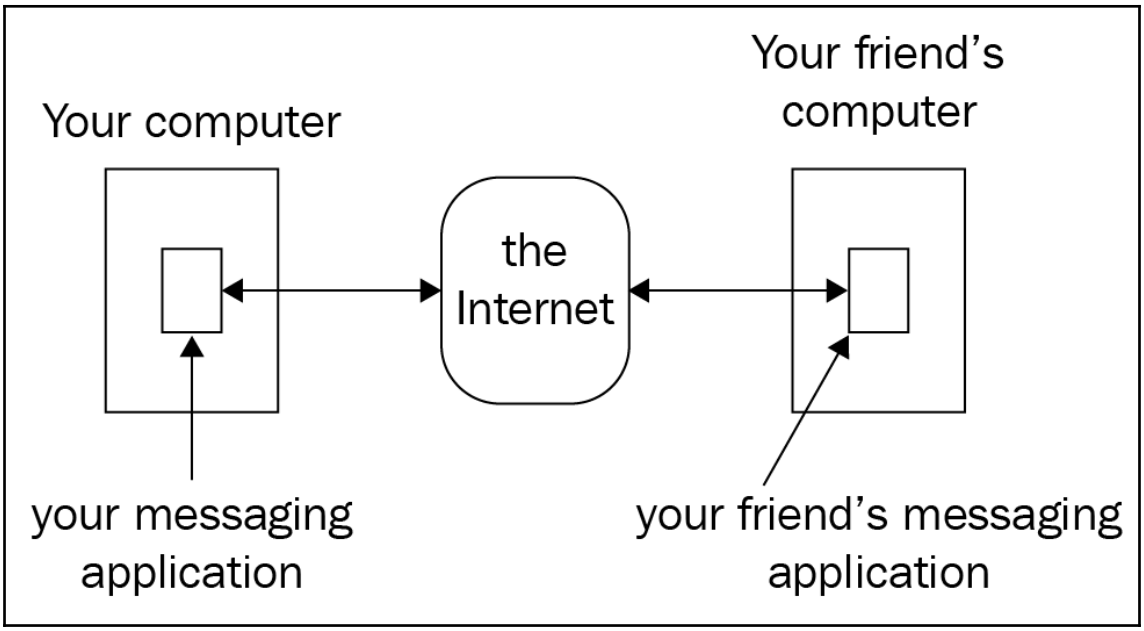

Mediator
- units: unordered_map<int, Unit*>
- buildings: unordered_map<int, Unit*>
+ notifyUnit(Unit*)
Player
- meditator : Mediator
Building
- mediator: Mediator
Player
- mediator: Mediator

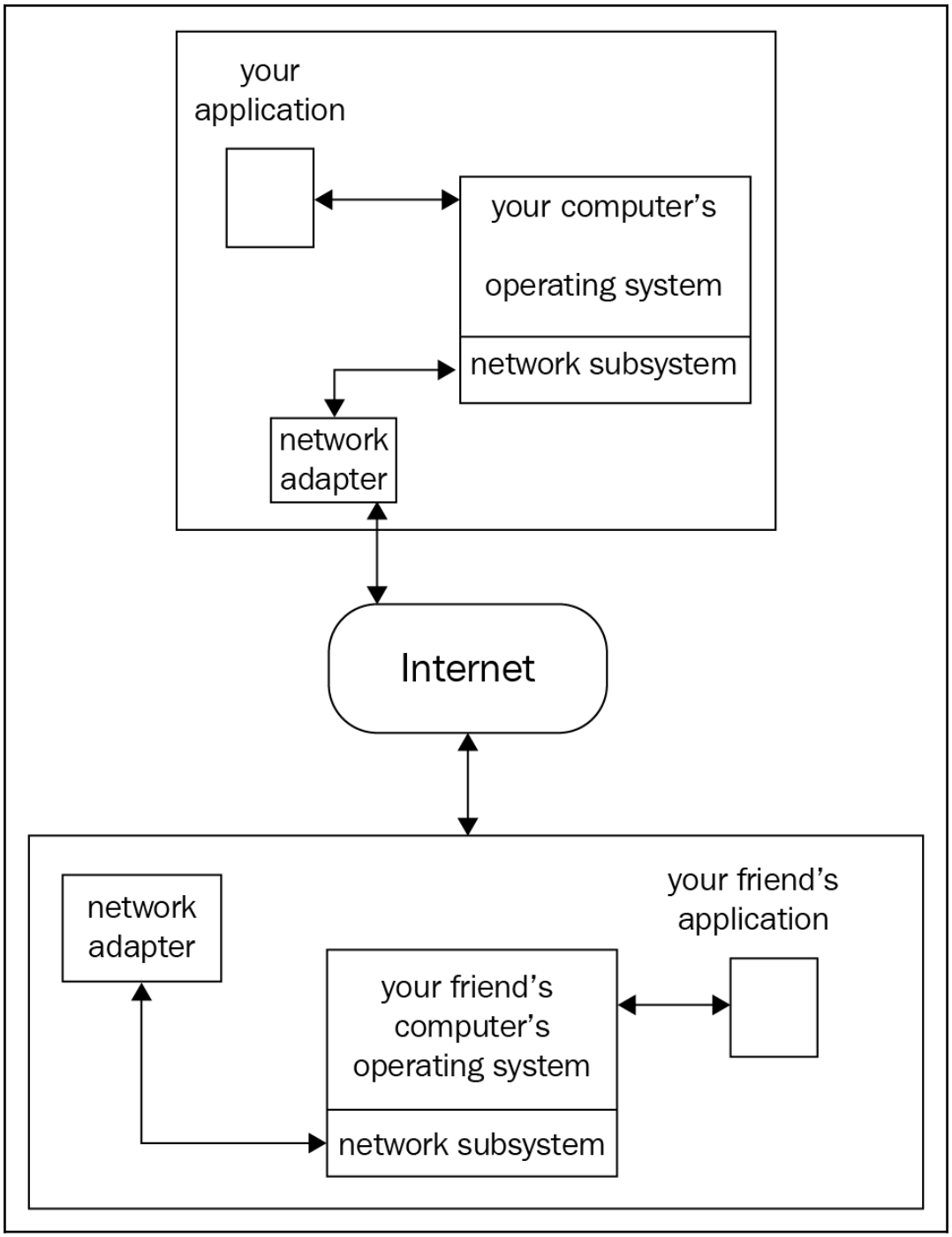

CharacterUnit
- height_ : int
- width_ : int
- sprite_: Image
- coords_ : Point
- power_ : int
- life_points_ : int
+ move(Point) : void
+ get_height() : int
...
CharacterUnit
- unit_data_ : UnitData*
- coords_ : Point
- power_ : int
- life_points_ : int
+ move(Point) : void
+ get_height() : int
...
UnitData
- height_ : int
- width_ : int
- sprite_ : Image
+ move(Point) : void
+ get_height() : int
...

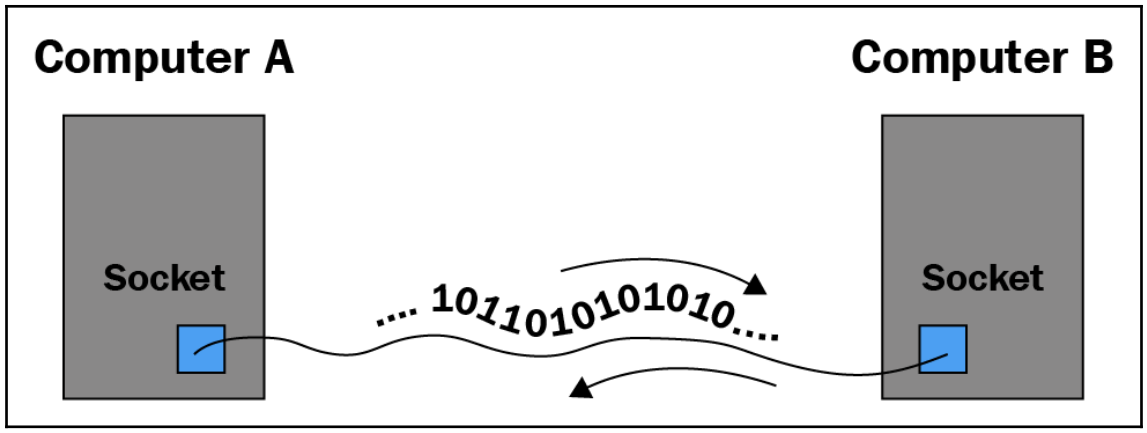

process user actions
update the game
wait N milliseconds

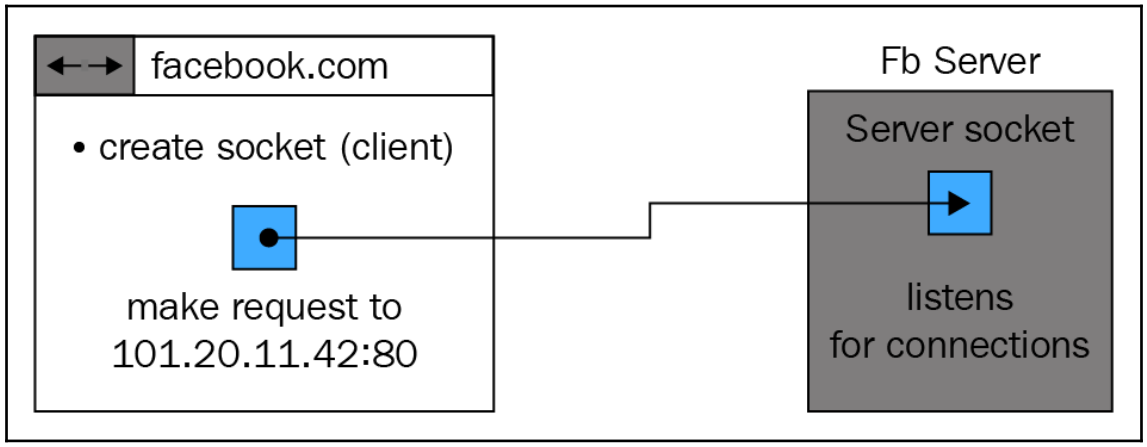
Chapter 12: Networking and Security

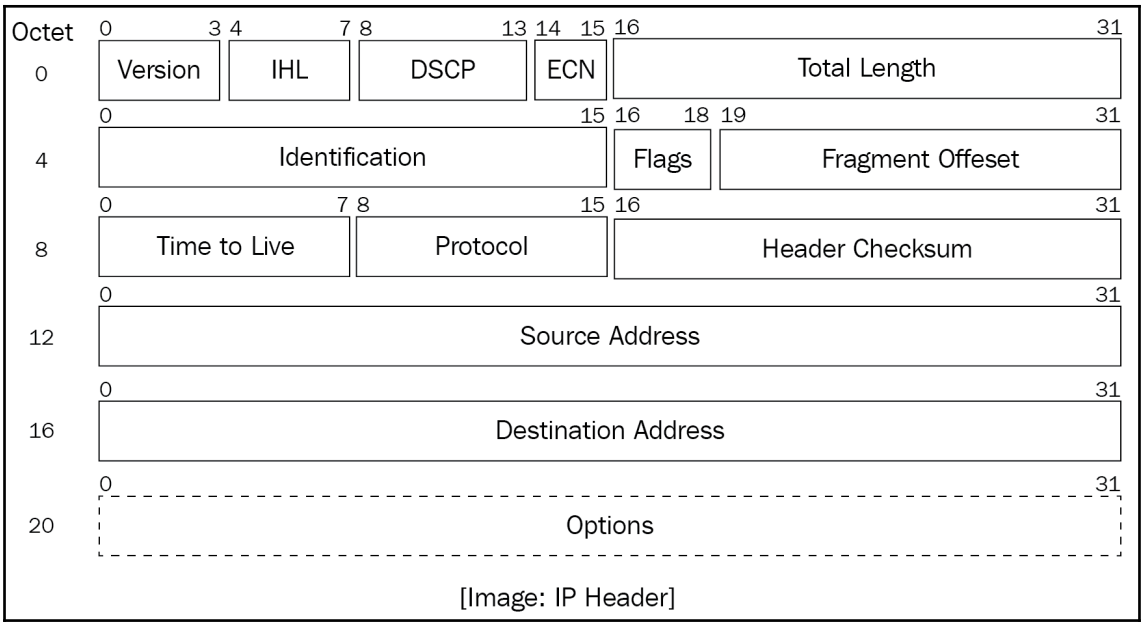



Your computer
Your friend's computer
the Internet
your messaging application
your friend's messaging application

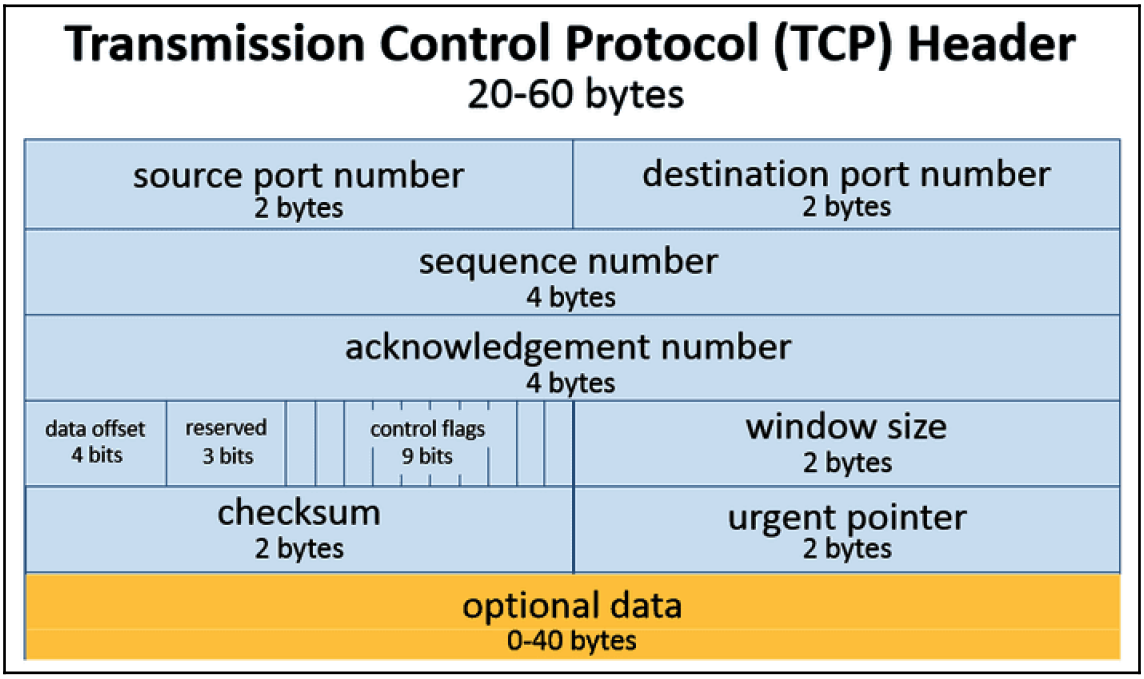

your
application
your computer's
operating system
network subsystem
network
adapter
Internet
network
adapter
your friend's
application
your friend's
computer's
operating system
network subsystem

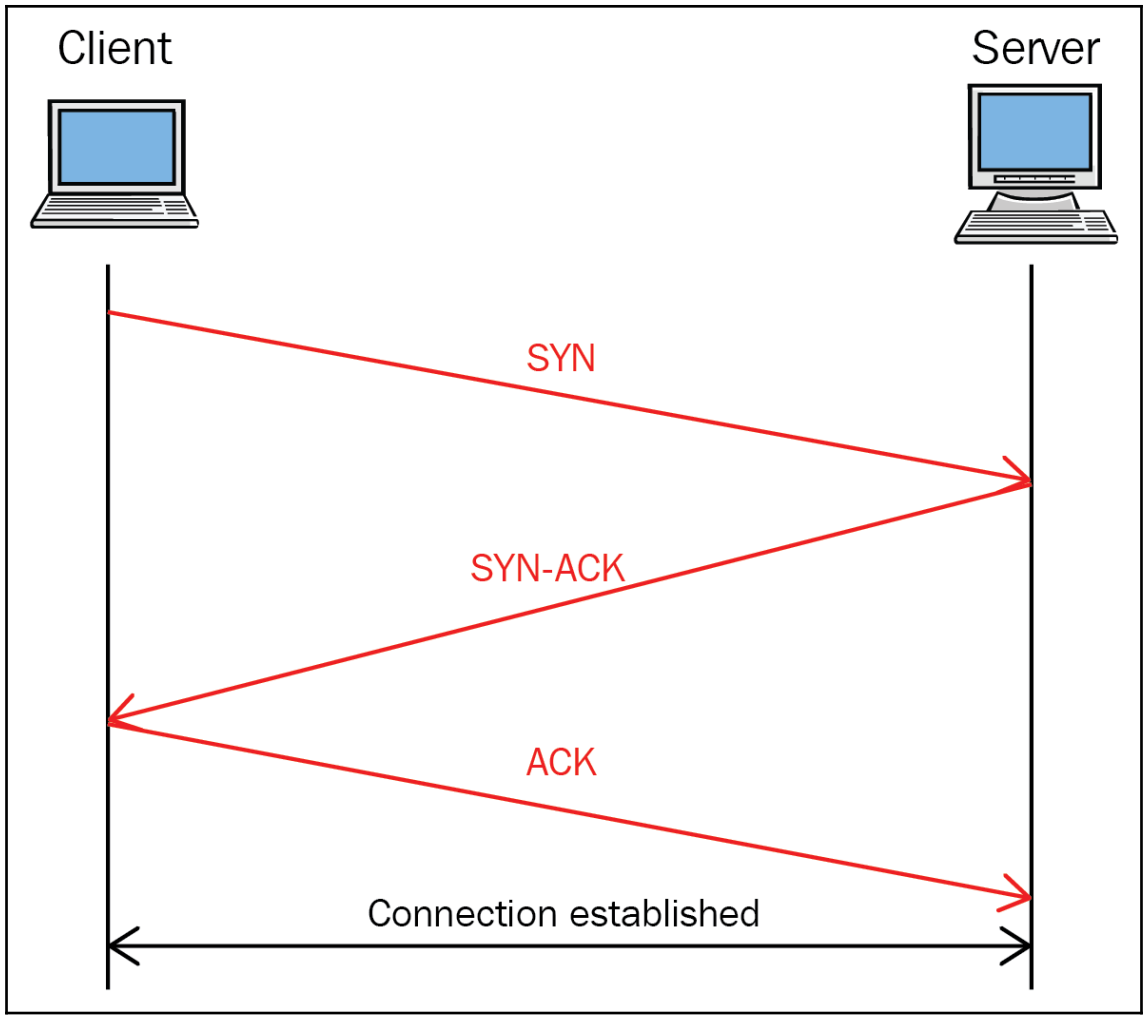

Computer A
Computer B
Socket
.... 1011010101010....
Socket

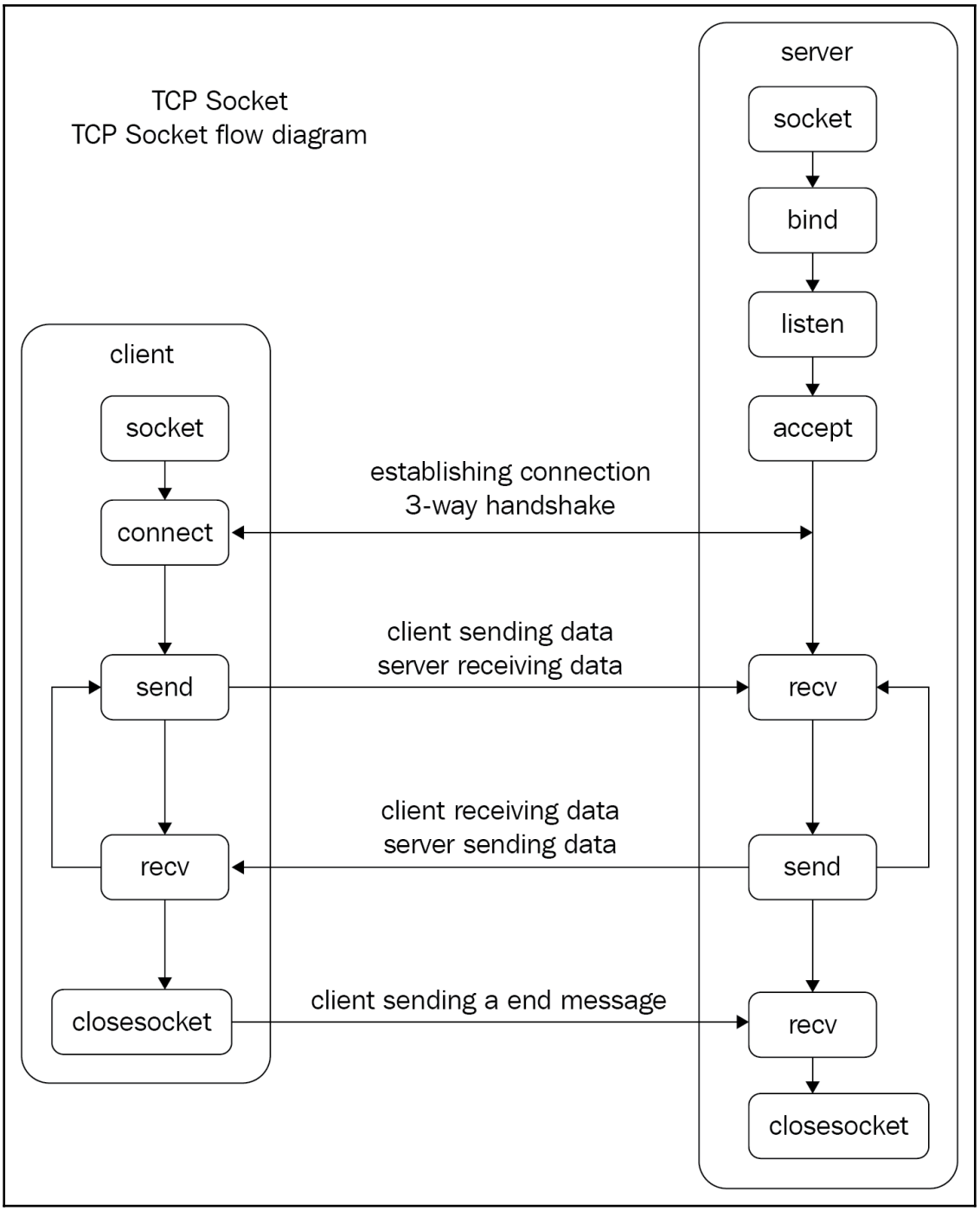

facebook.com
• create socket (client)
make request to
101.20.11.42:80
Fb Server
Server socket
listens
for connections

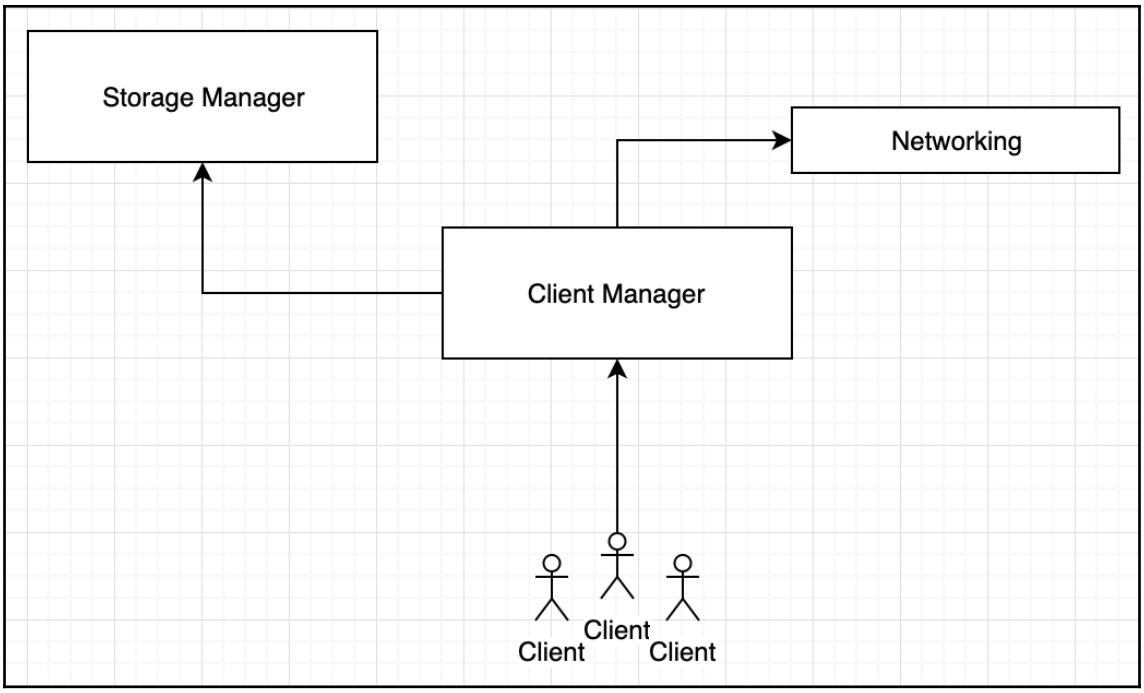


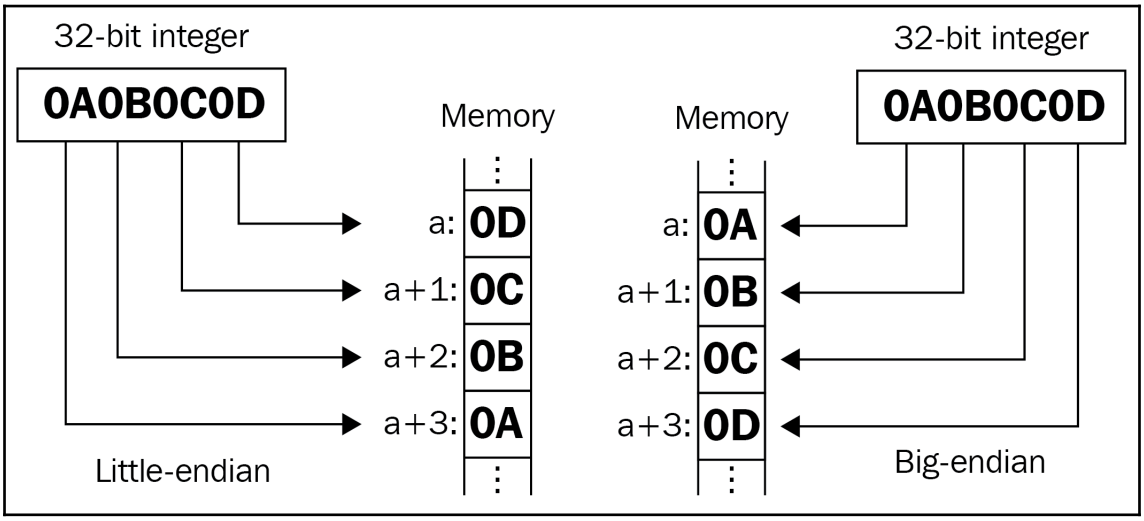
[Image: IP Header]

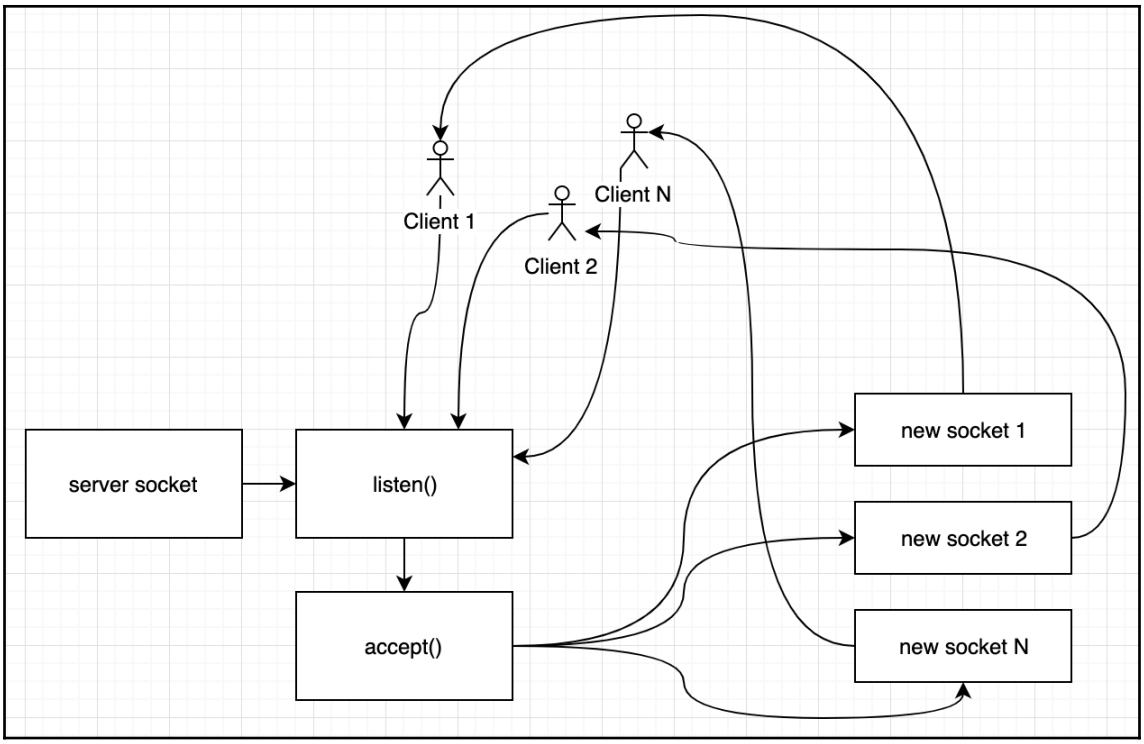


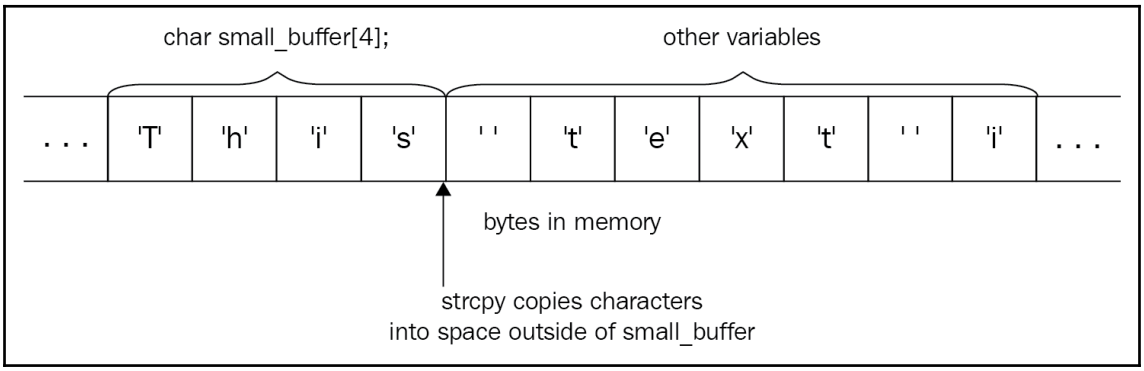

Client
Server
SYN
SYN-ACK
ACK
Connection established


TCP Socket
TCP Socket flow diagram
server
socket
bind
listen
accept
client
socket
connect
send
recv
closesocket
establishing connection
3-way handshake
client sending data
server receiving data
client receiving data
server sending data
client sending a end message
recv
send
recv
closesocket

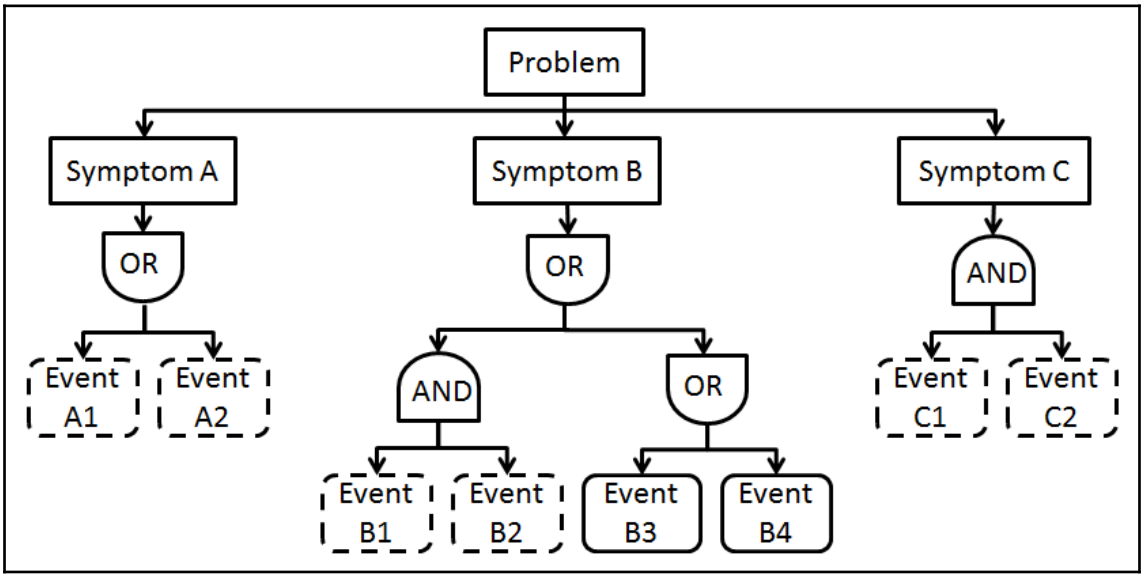

Storage Manager
Networking
Client Manager
Client
Client
Client

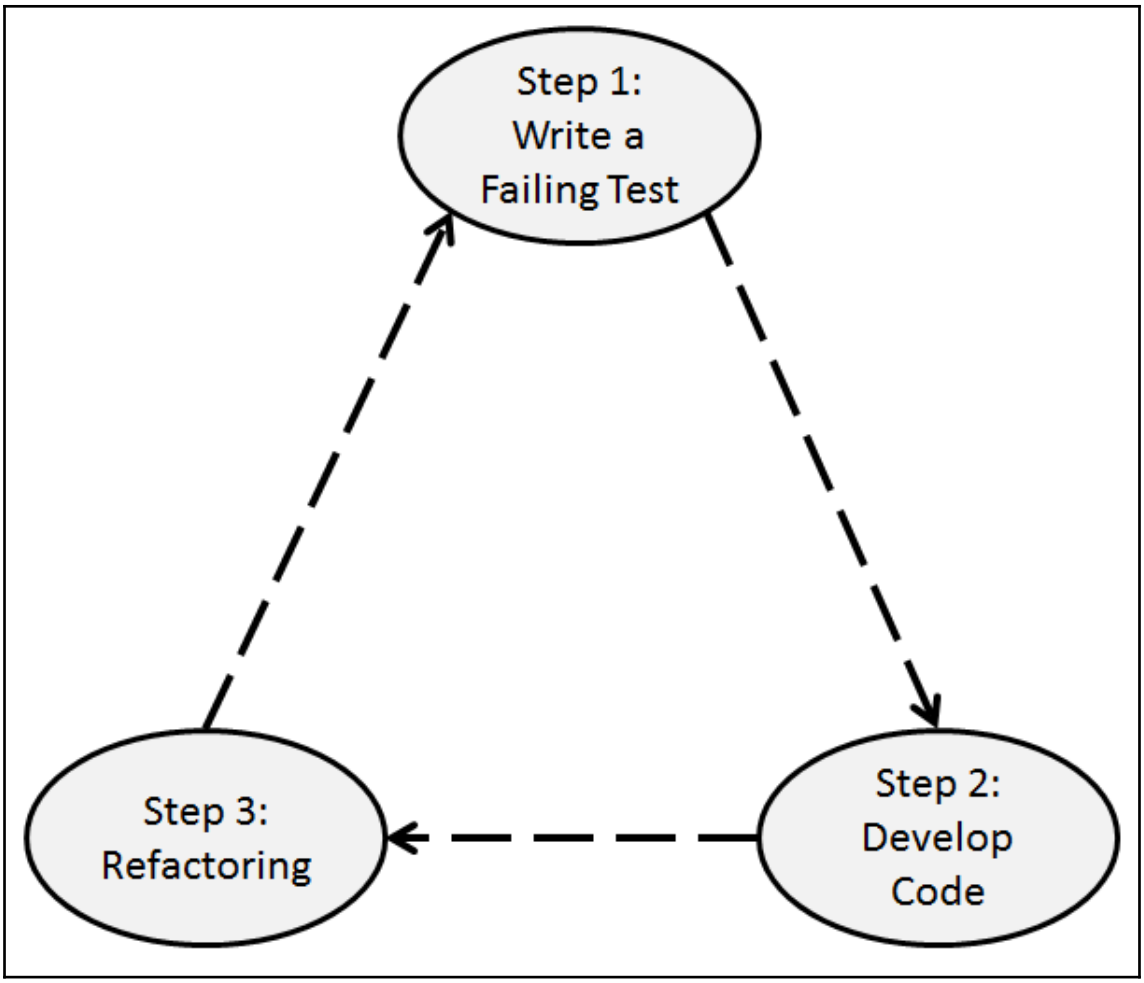

32-bit integer
0A0B0C0D
Memory
a: 0D
a+1: 0C
a+2: 0B
a+3: 0A
Little-endian
Memory
a: 0A
a+1: 0B
a+2: 0C
a+3: 0D
32-bit integer
0A0B0C0D
Big-endian

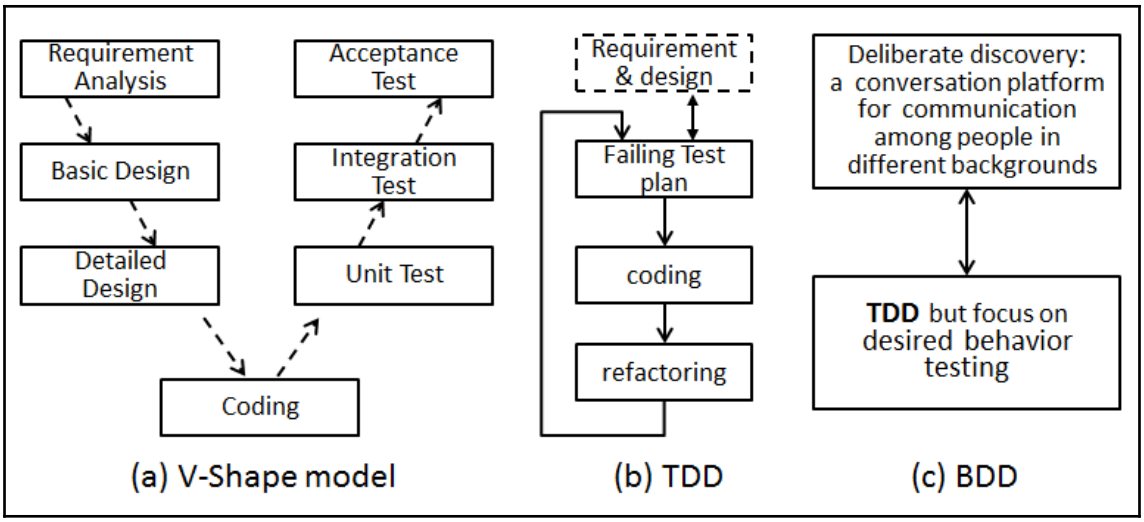

Client 1
Client 2
Client N
server socket
listen()
accept()
new socket 1
new socket 2
new socket N


char small_buffer[4];
other variables
. . .
'T'
'h'
'i'
's'
' '
't'
'e'
'x'
't'
' '
'i'
. . .
bytes in memory
strcpy copies characters
into space outside of small_buffer

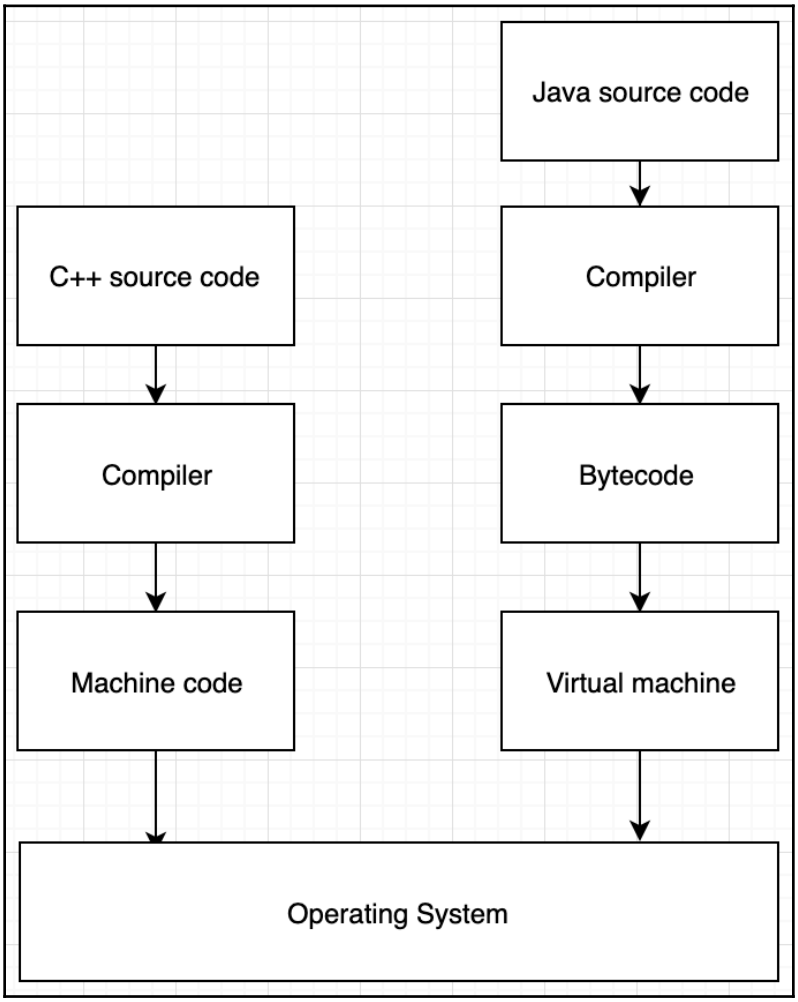
Chapter 13: Debugging and Testing

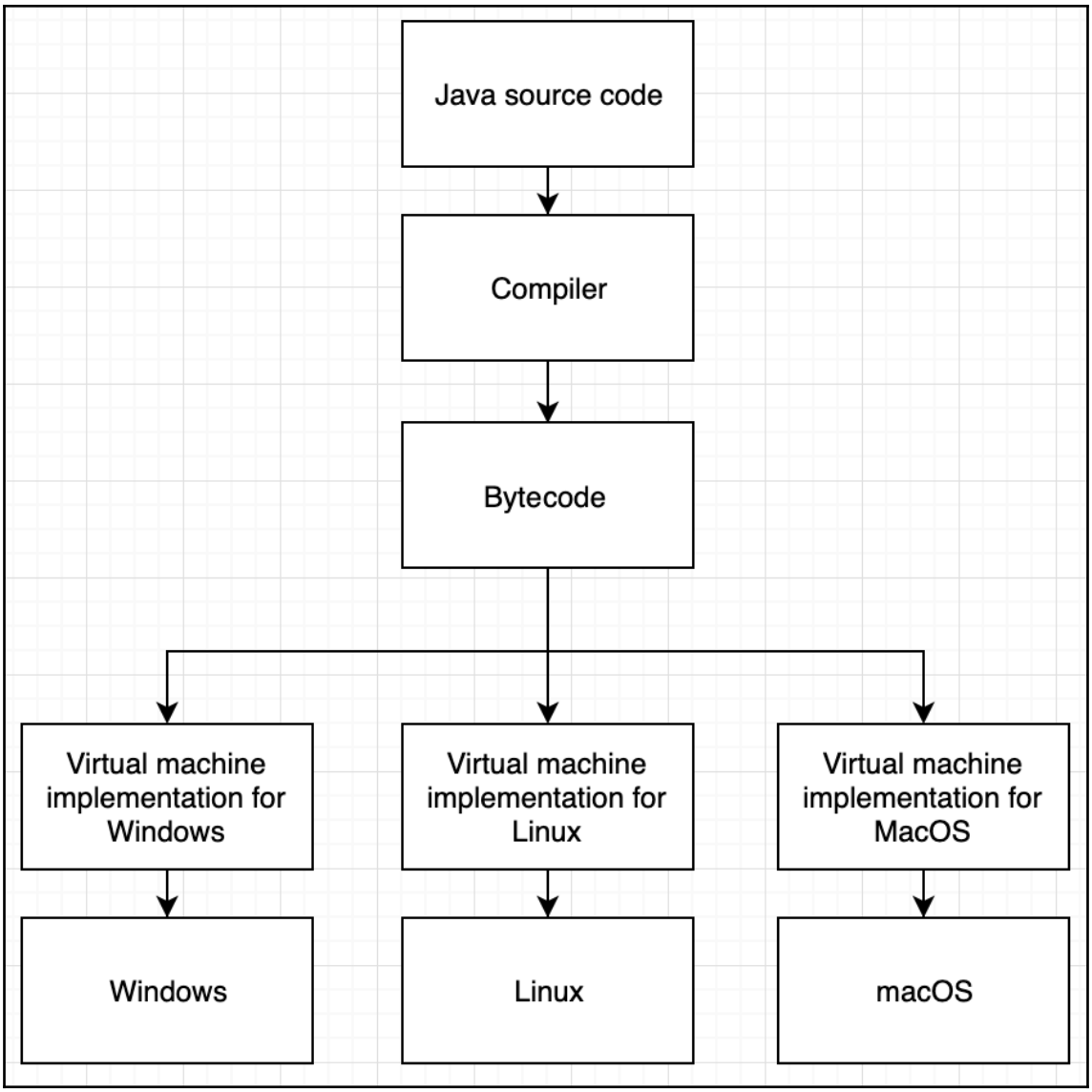


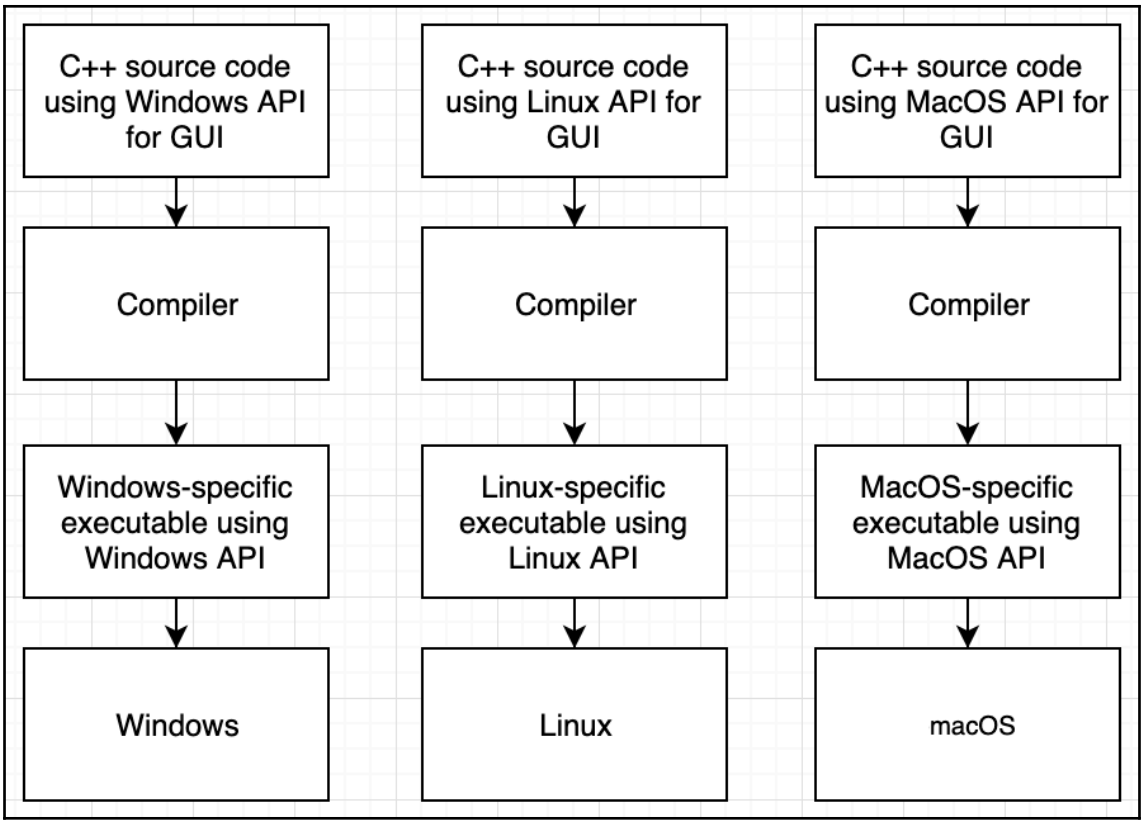

Step 1:
Write a
Failing Test
Step 2:
Develop
Code
Step 3:
Refactoring

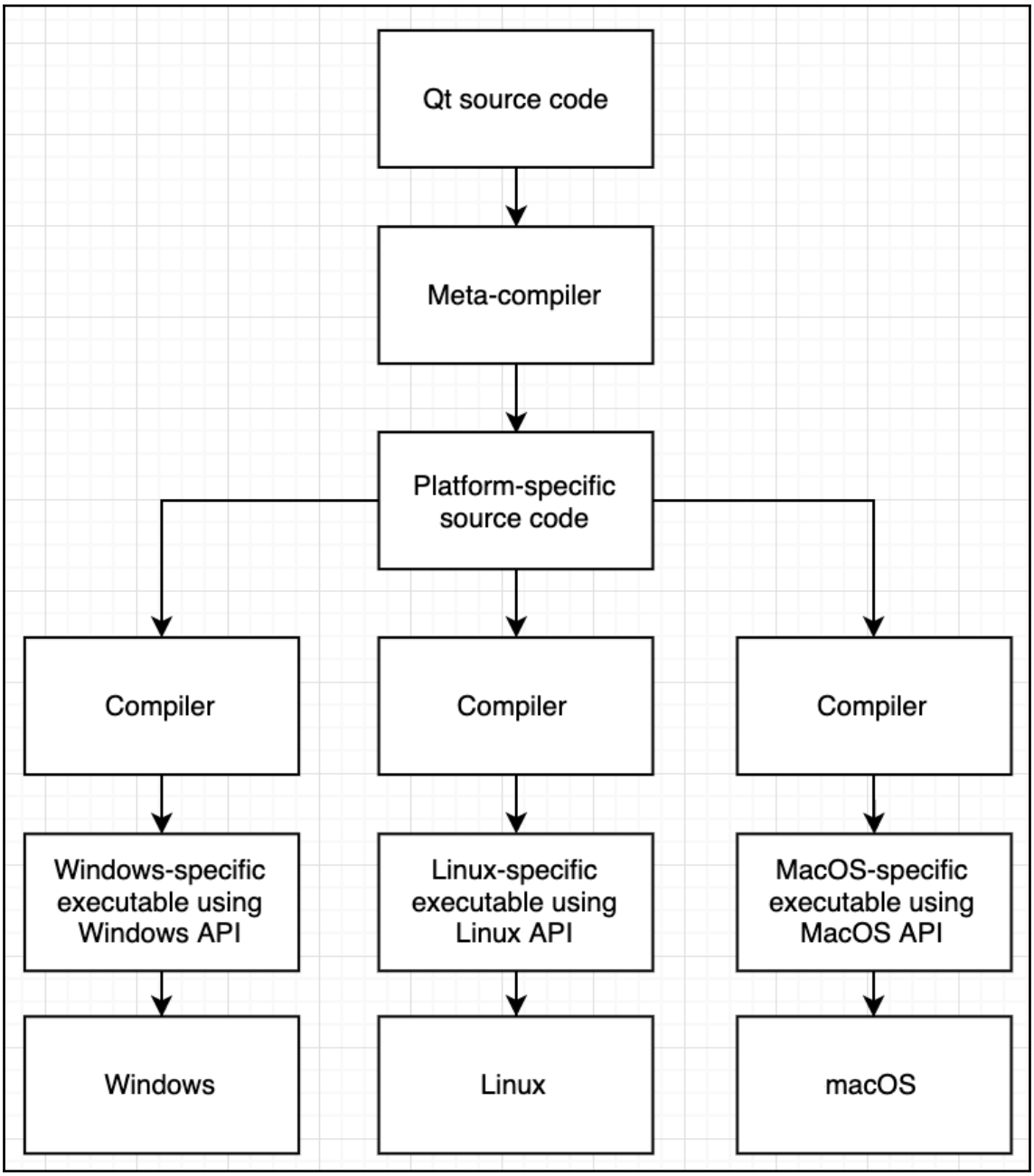

Requirement Analysis
Basic Design
Detailed Design
Coding
Acceptance Test
Integration Test
Unit Test
(a) V-Shape model
Requirement & design
Failing Test plan
coding
refactoring
(b) TDD
Deliberate discovery: a conversation platform for communication among people in different backgrounds
TDD but focus on desired behavior testing
(c) BDD

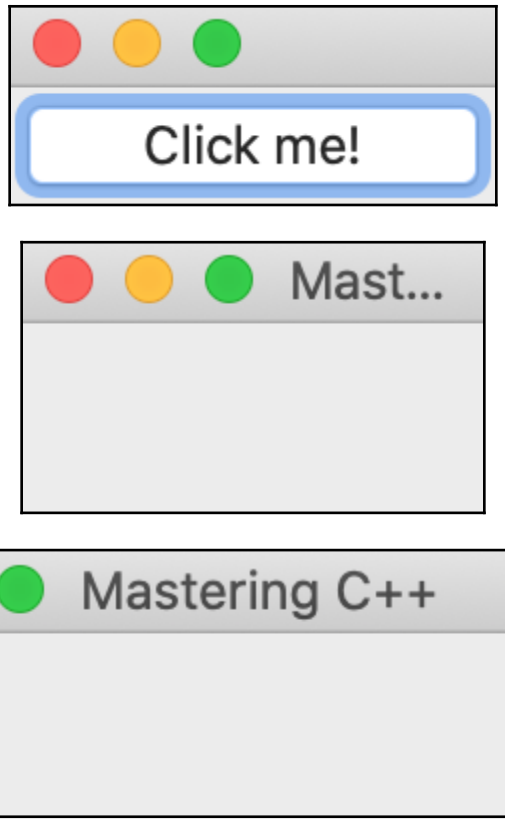
Chapter 14: Graphical User Interface with Qt

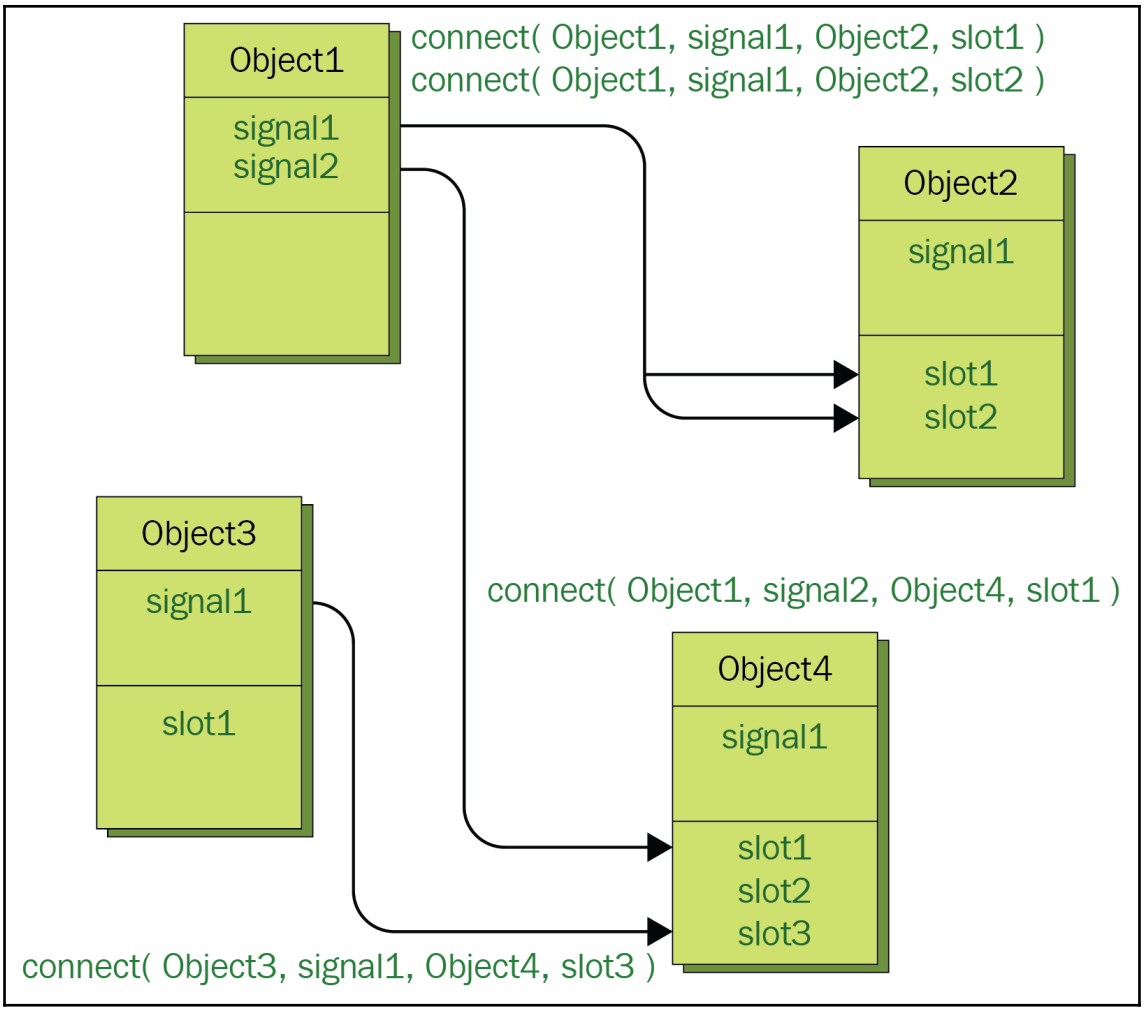


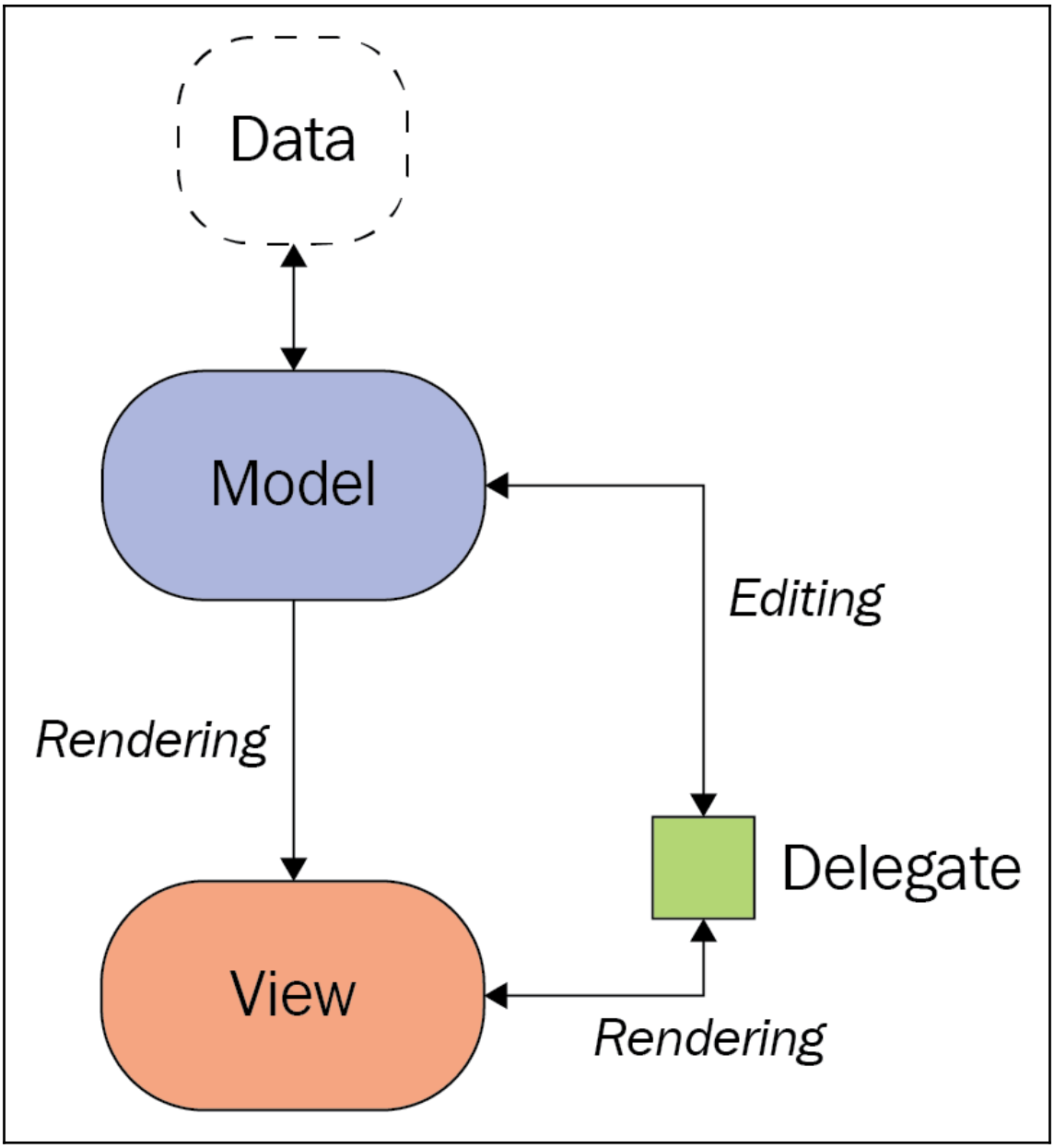

Java source code
Compiler
Bytecode
Virtual machine implementation for Windows
Virtual machine implementation for Linux
Virtual machine implementation for MacOS
Windows
Linux
macOS

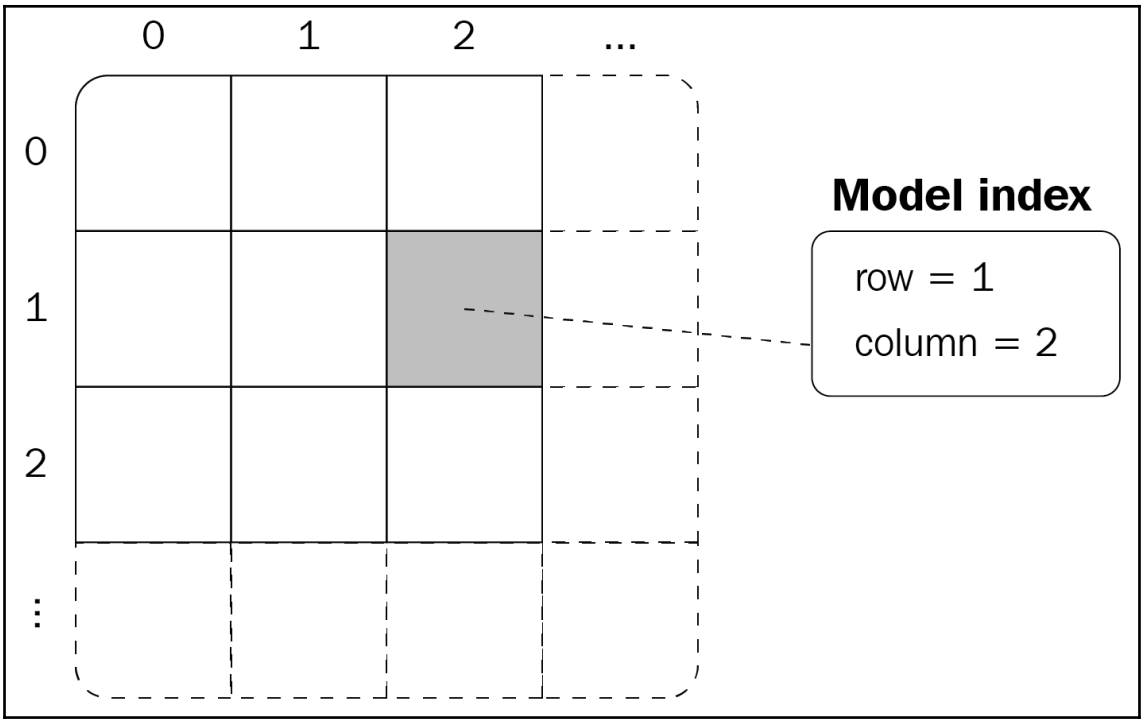

C++ source code using Windows API for GUI
C++ source code using Linux API for GUI
C++ source code using MacOS API for GUI
Compiler
Compiler
Compiler
Windows-specific executable using Windows API
Linux-specific executable using Linux API
MacOS-specific executable using MacOS API
Windows
Linux
macOS

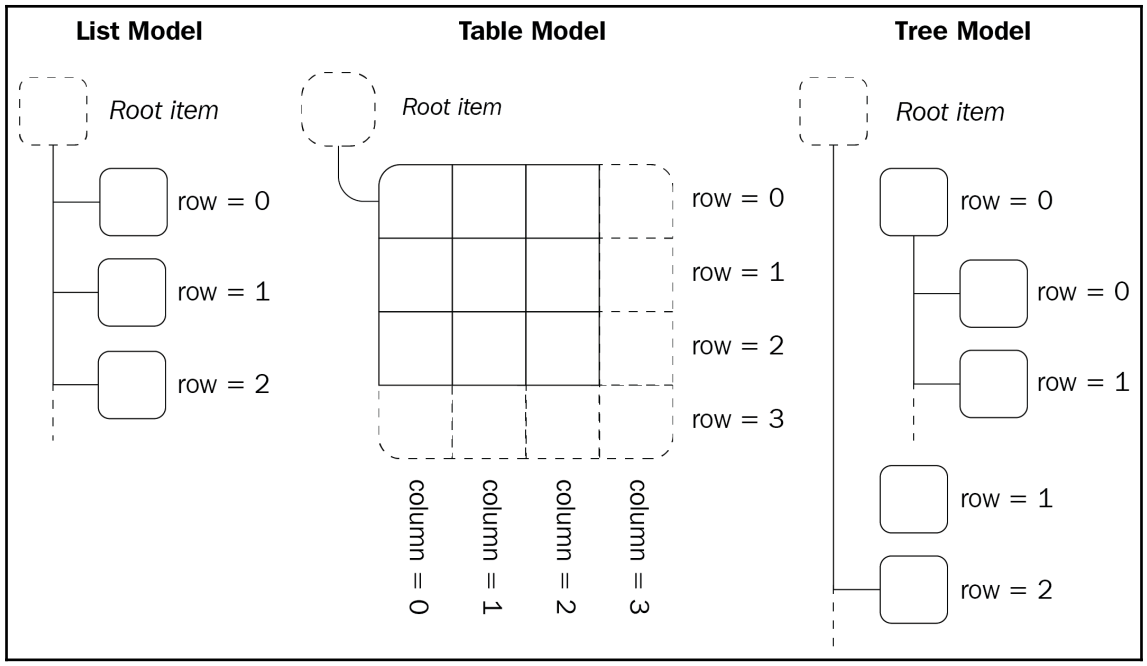

Qt source code
Meta-compiler
Platform-specific source code
Compiler
Compiler
Compiler
Windows-specific executable using Windows API
Linux-specific executable using Linux API
MacOS-specific executable using MacOS API
Windows
Linux
macOS

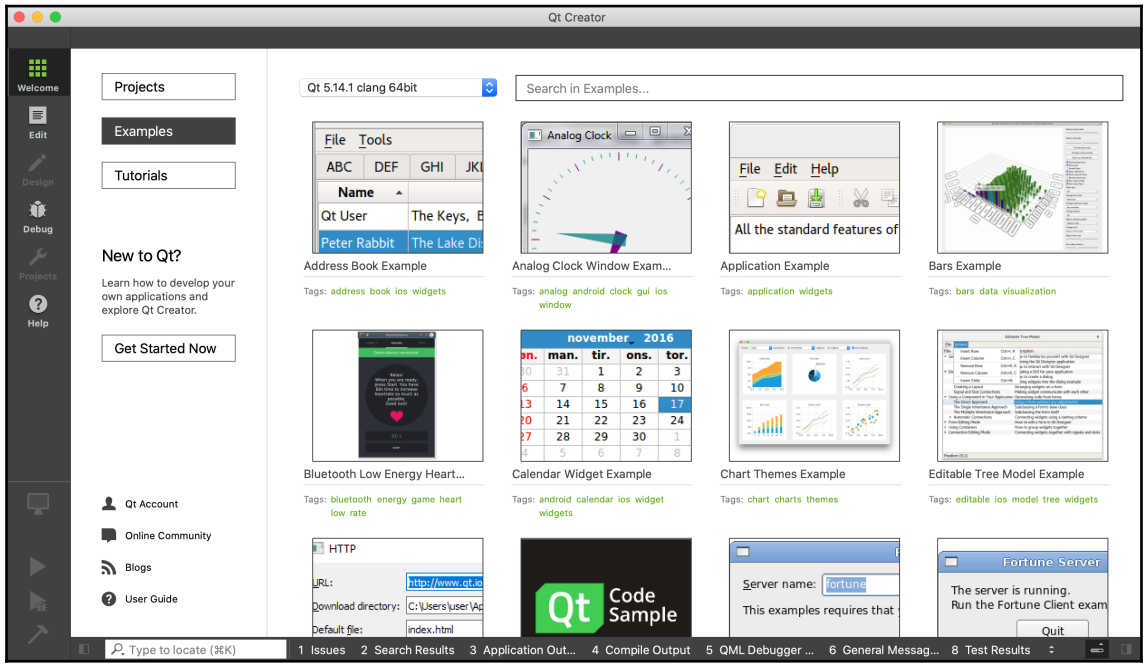

Click me!
Mast...
Mastering C++

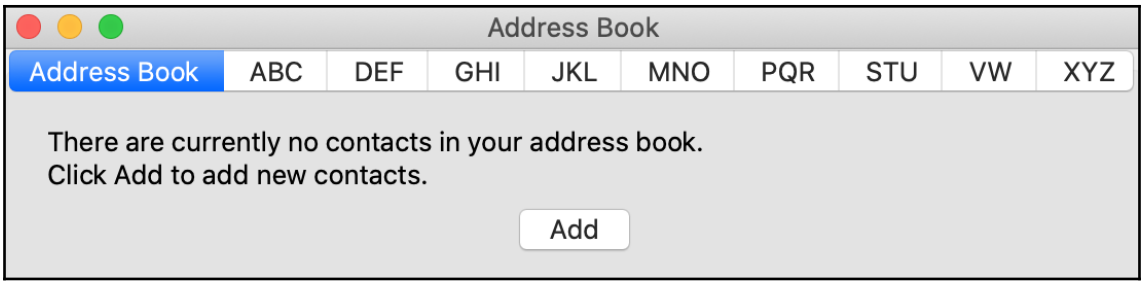

Object1
signal1
signal2
connect(Object1, signal1, Object2, slot1)
connect(Object1, signal1, Object2, slot2)
Object2
signal1
slot1
slot2
Object3
signal1
slot1
connect(Object1, signal2, Object4, slot1)
Object4
signal1
slot1
slot2
slot3
connect(Object3, signal1, Object4, slot3)

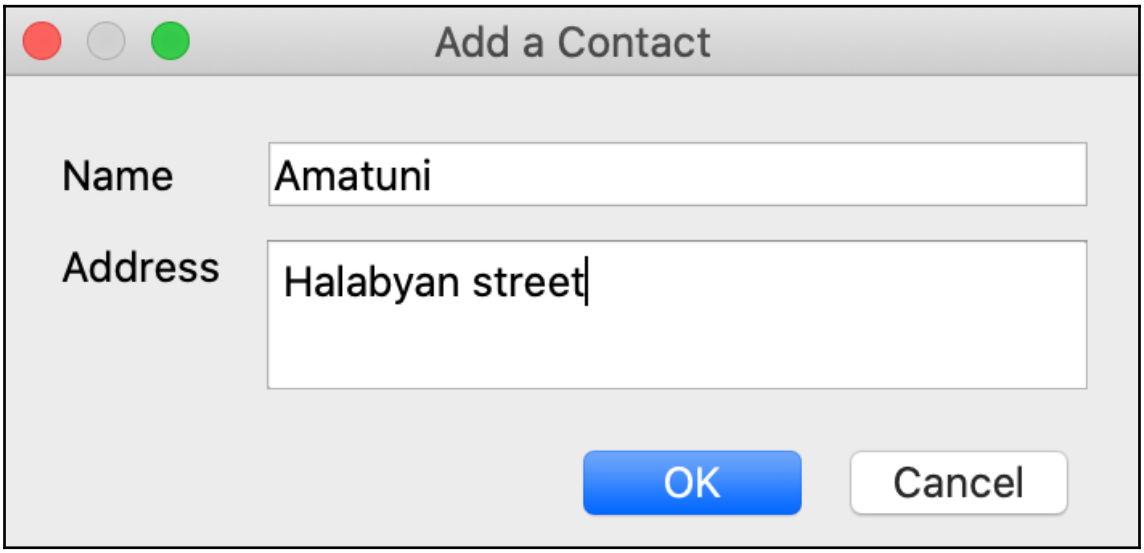

Data
Model
Editing
Rendering
Delegate
View
Rendering

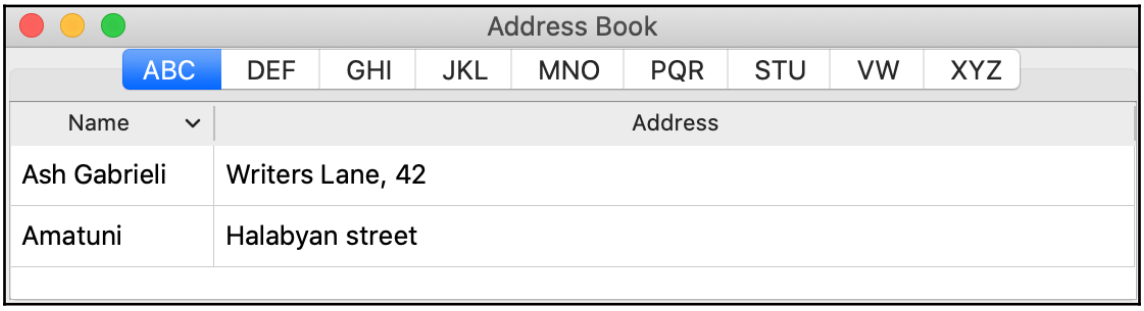

0
1
2
...
0
1
2
⋮
Model index
row = 1
column = 2


List Model
Root item
row = 0
row = 1
row = 2
Table Model
Root item
row = 0
row = 1
row = 2
row = 3
column = 0
column = 1
column = 2
column = 3
Tree Model
Root item
row = 0
row = 0
row = 1
row = 1
row = 2

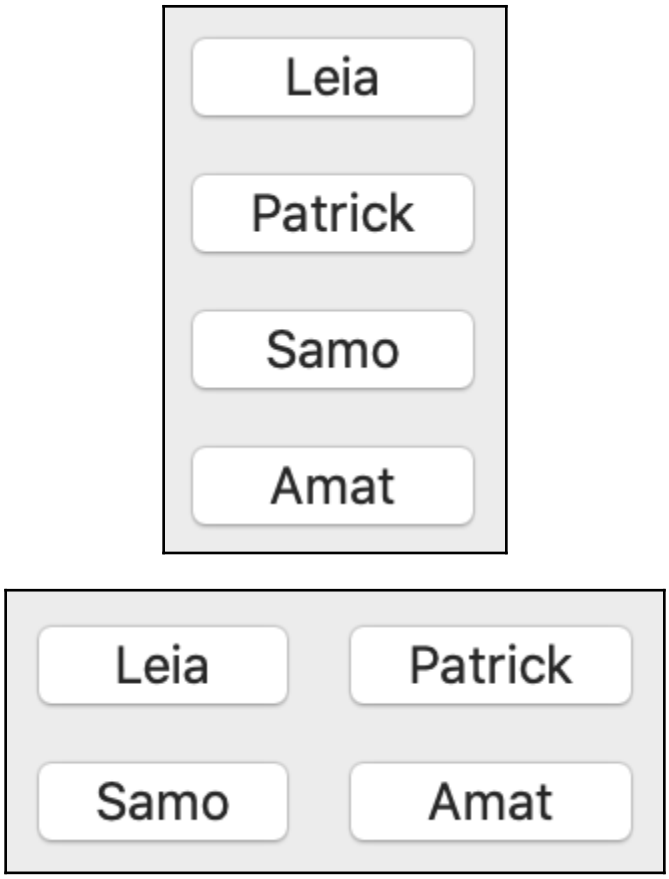

Qt Creator
Welcome
Edit
Design
Debug
Projects
Help
Projects
Examples
Tutorials
New to Qt?
Learn how to develop your own applications and explore Qt Creator.
Get Started Now
Qt Account
Online Community
Blogs
User Guide
Qt 5.14.1 clang 64bit
Search in Examples...
Address Book Example
Analog Clock Window Exam...
Application Example
Bars Example
Bluetooth Low Energy Heart...
Calendar Widget Example
Chart Themes Example
Editable Tree Model Example
Code Sample
Fortune Server
Type to locate (⌘K)
1 Issues
2 Search Results
3 Application Out...
4 Compile Output
5 QML Debugger ...
6 General Messag...
8 Test Results


Address Book
Address Book
ABC
DEF
GHI
JKL
MNO
PQR
STU
VW
XYZ
There are currently no contacts in your address book.
Click Add to add new contacts.
Add

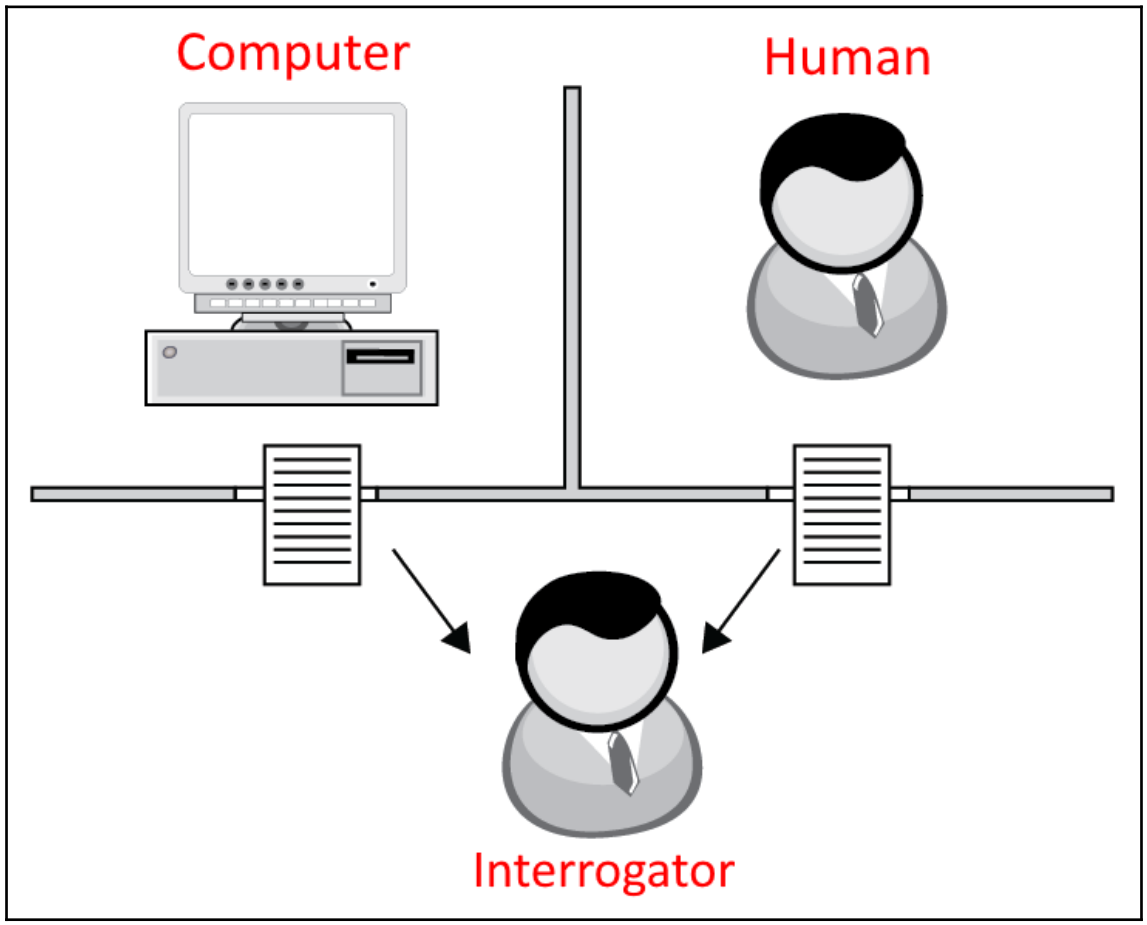

Add a Contact
Name
Amatuni
Address
Halabyan street
OK
Cancel

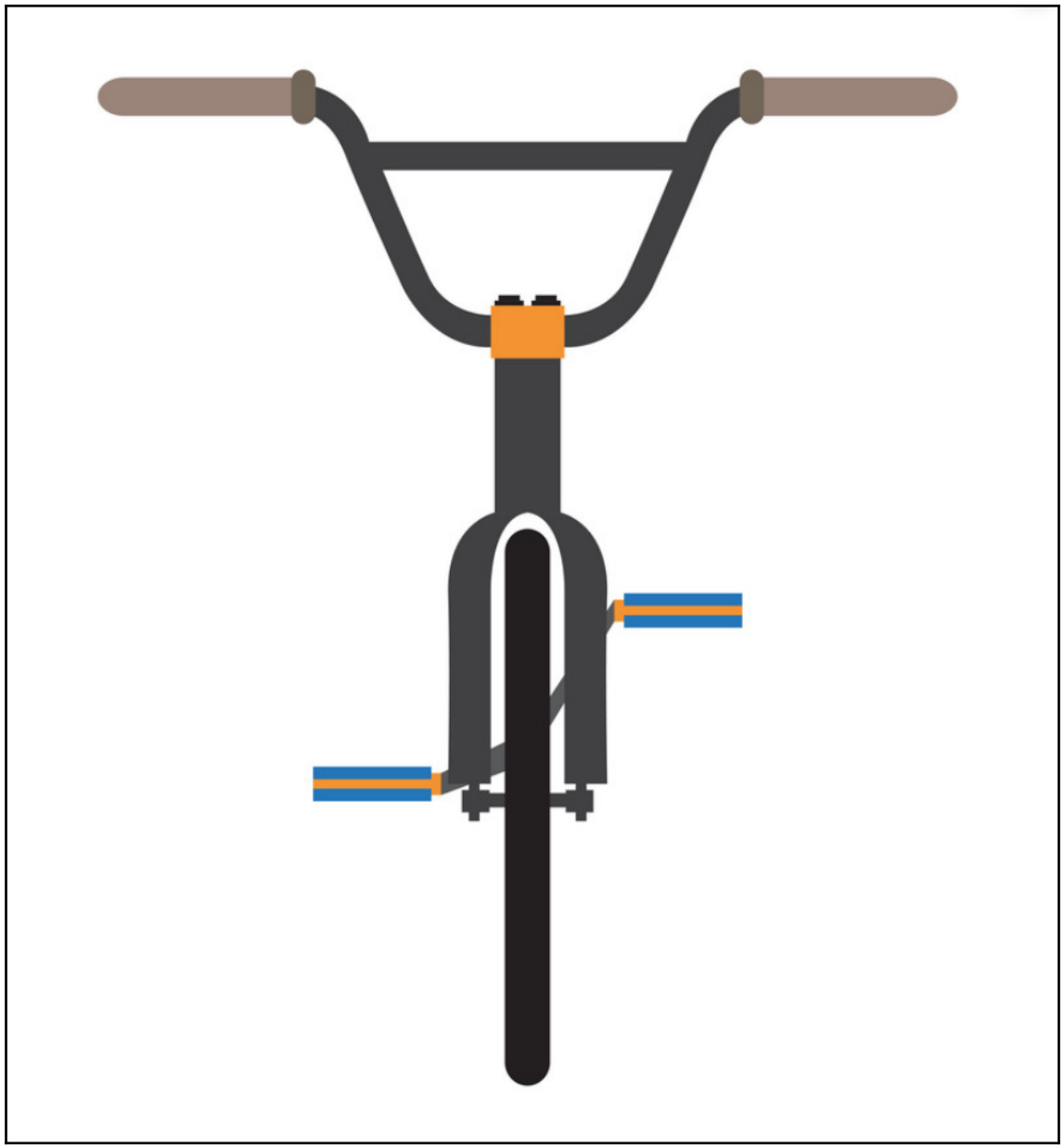

Address Book
ABC
DEF
GHI
JKL
MNO
PQR
STU
VW
XYZ
Name
Address
Ash Gabrieli
Writers Lane, 42
Amatuni
Halabyan street

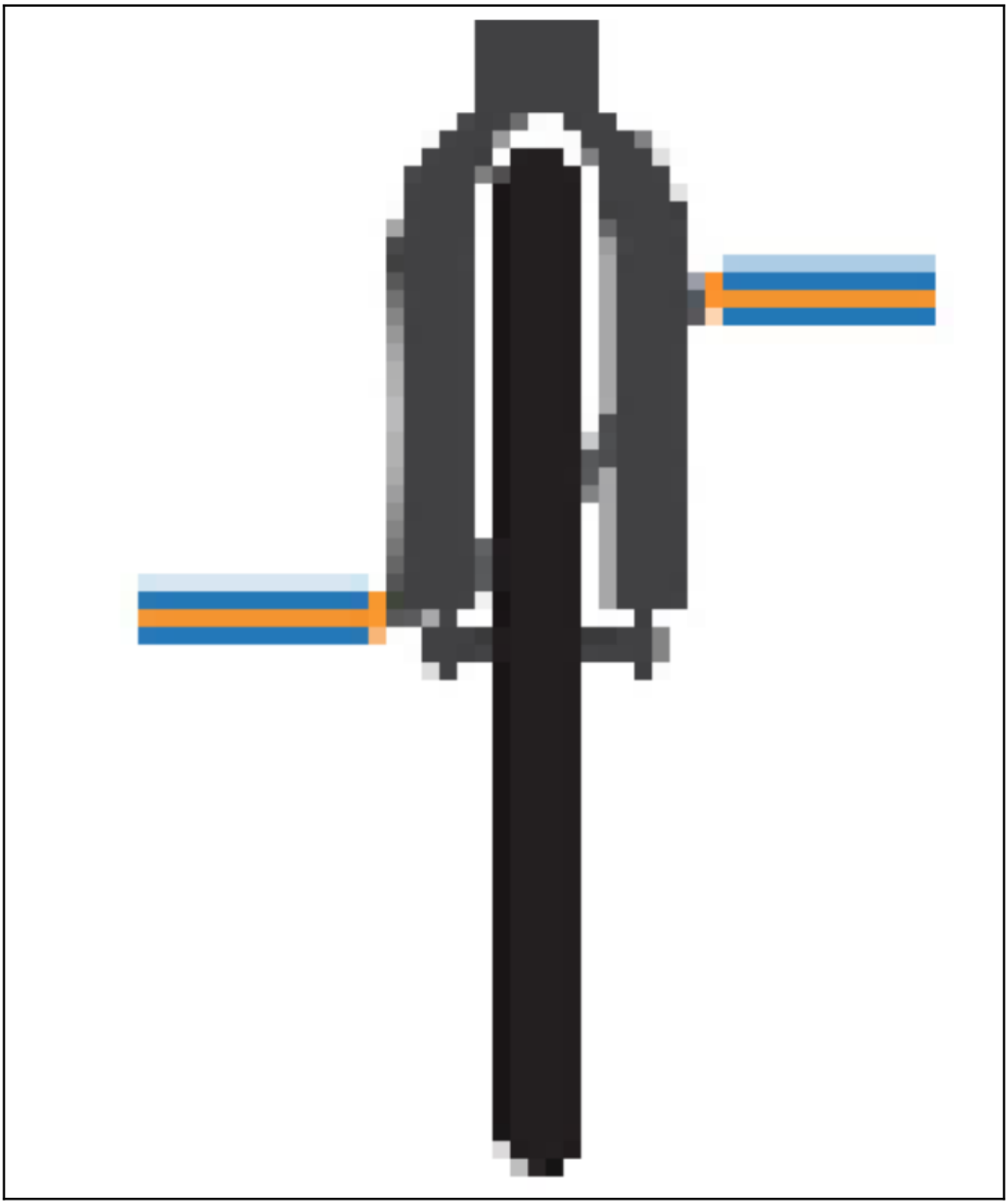

Leia
Patrick
Samo
Amat

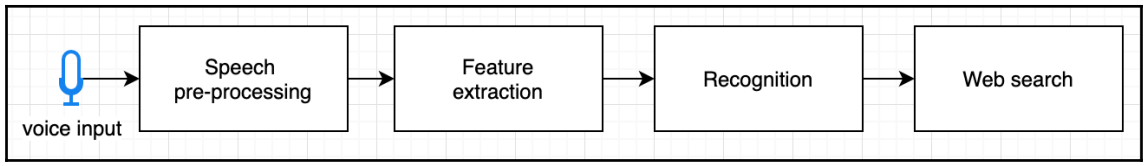

Leia
Patrick
Samo
Amat
Leia
Patrick
Samo
Amat

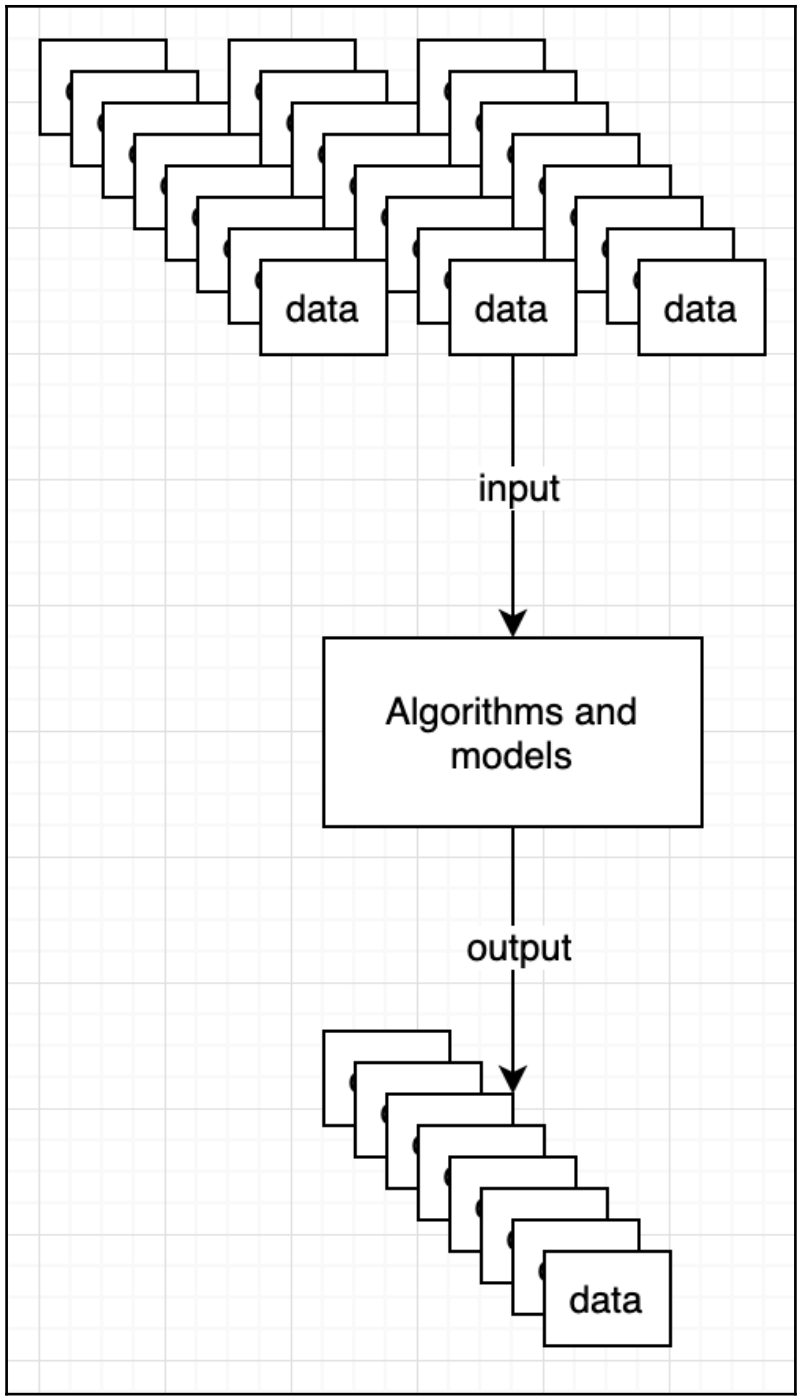
Chapter 15: Using C++ in Machine Learning Tasks

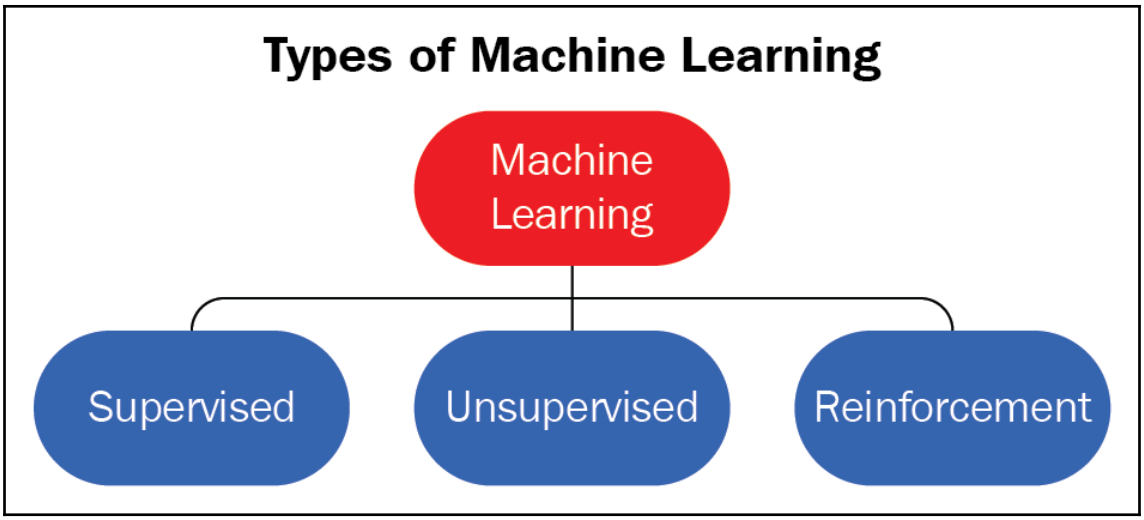


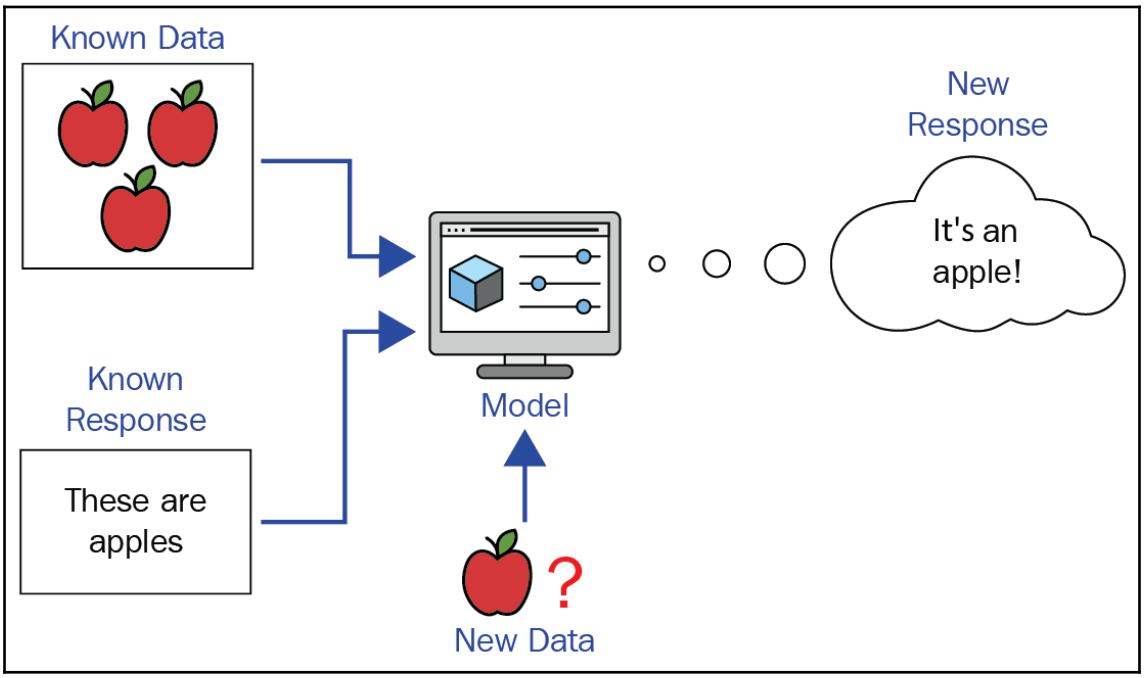


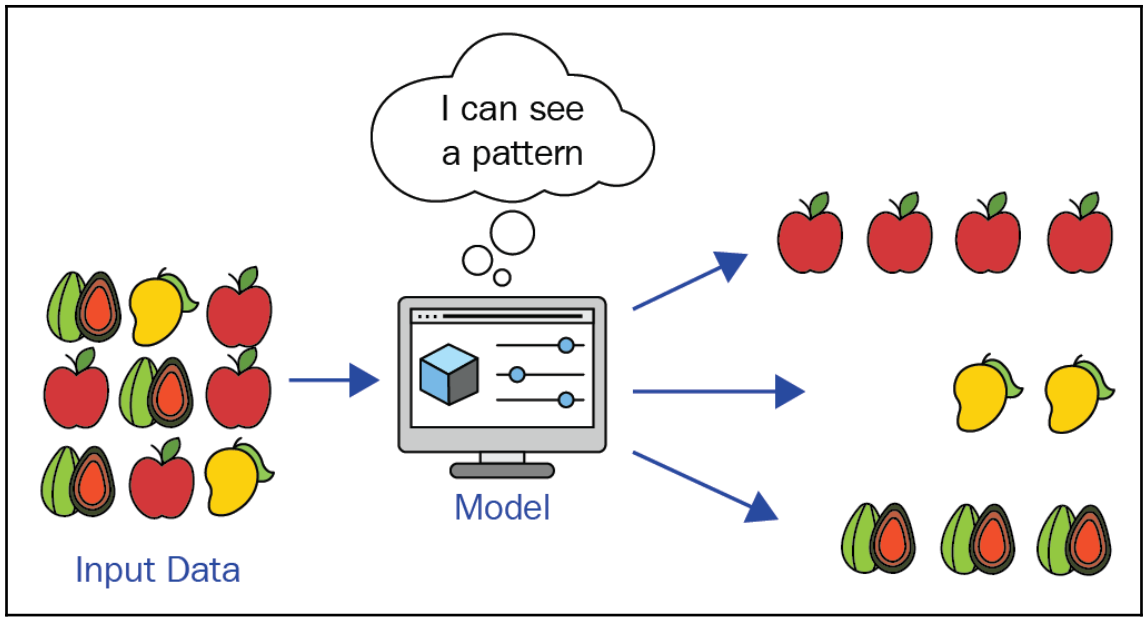


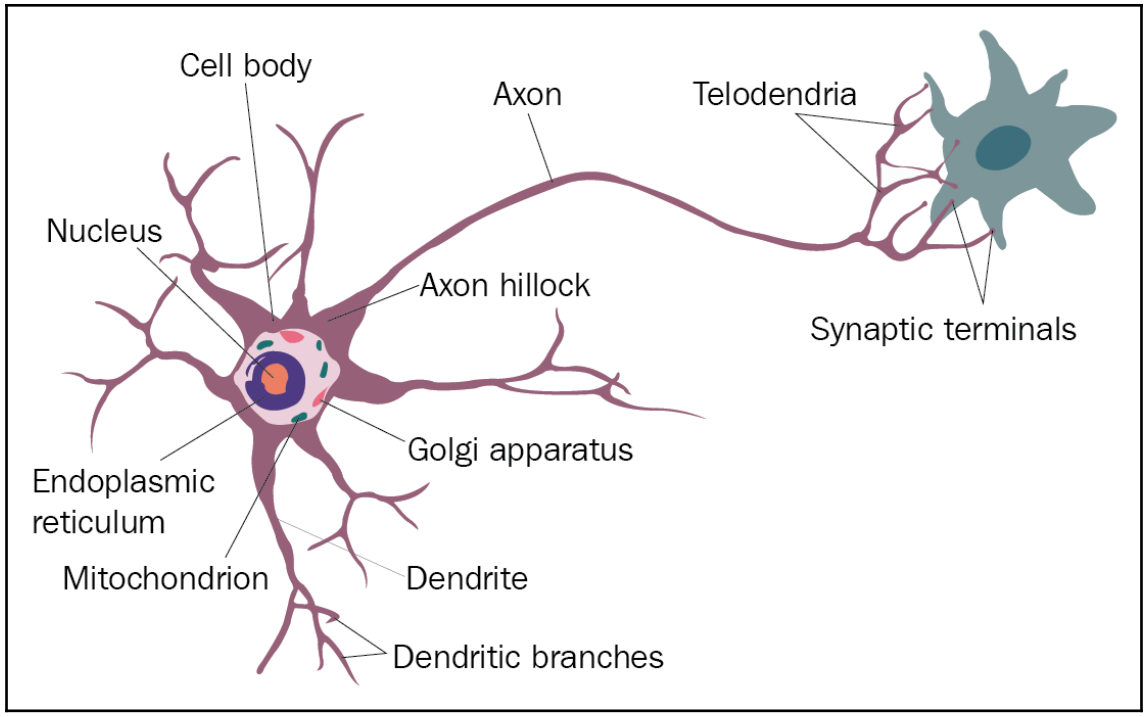

voice input
Speech pre-processing
Feature extraction
Recognition
Web search

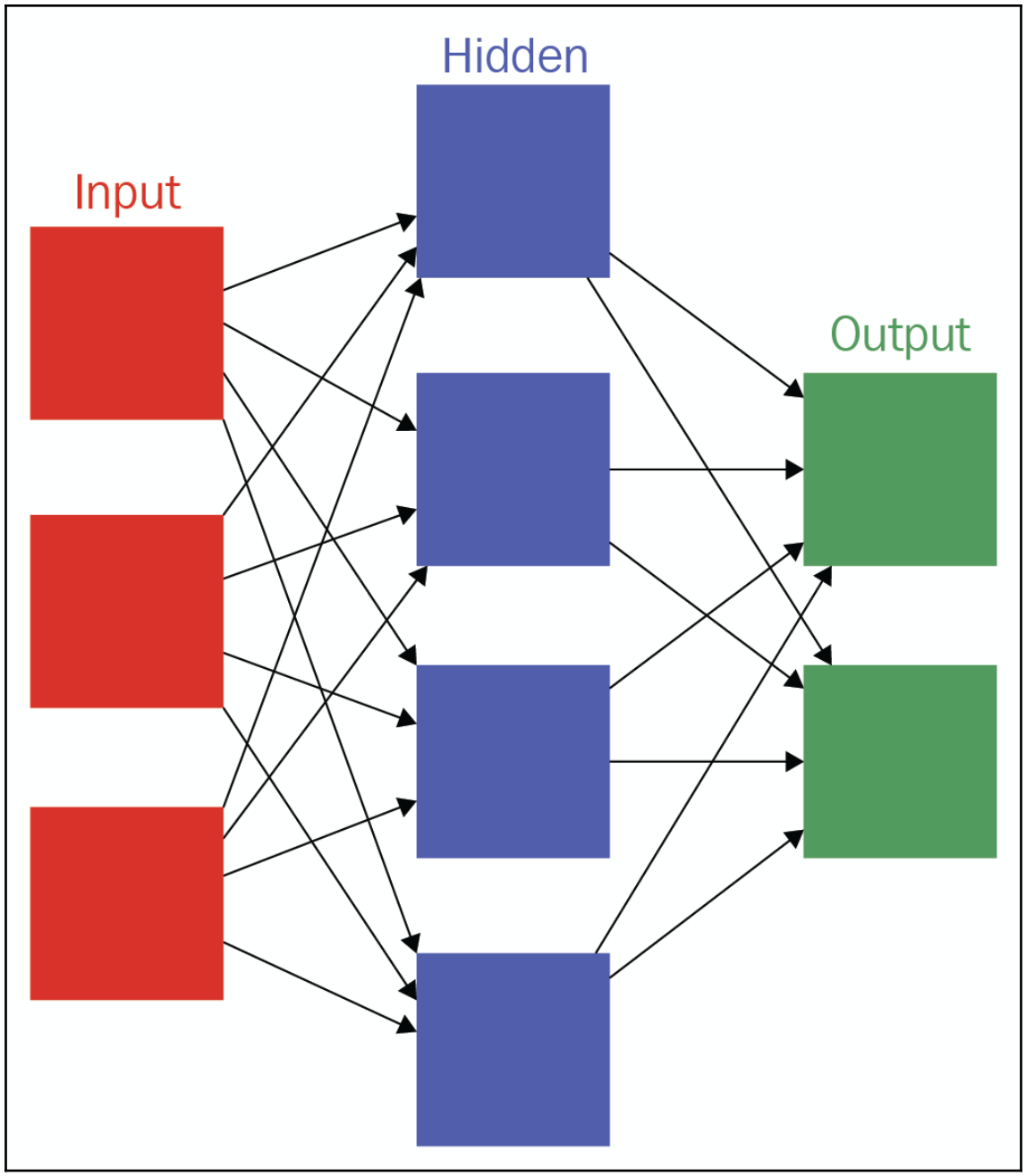

data
data
data
input
Algorithms and models
output
data

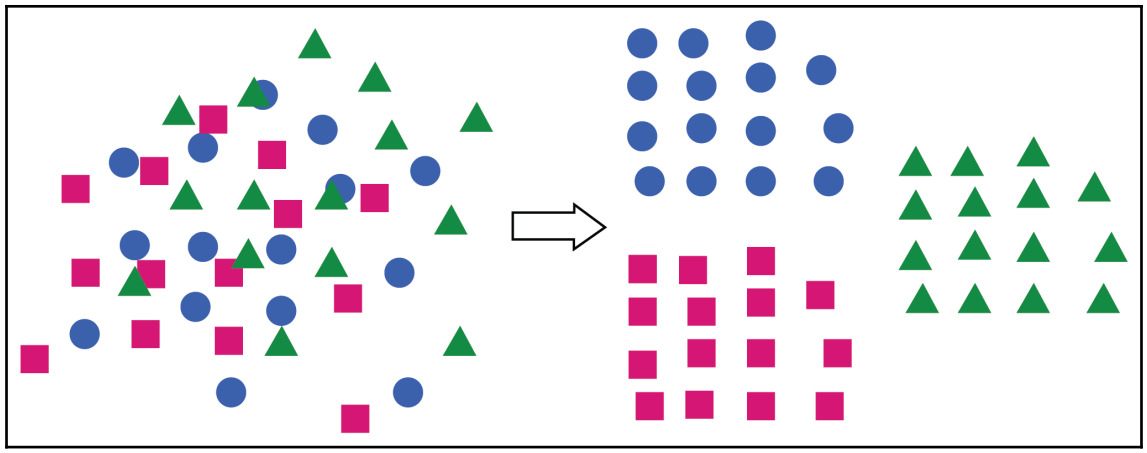

Types of Machine Learning
Machine Learning
Supervised
Unsupervised
Reinforcement

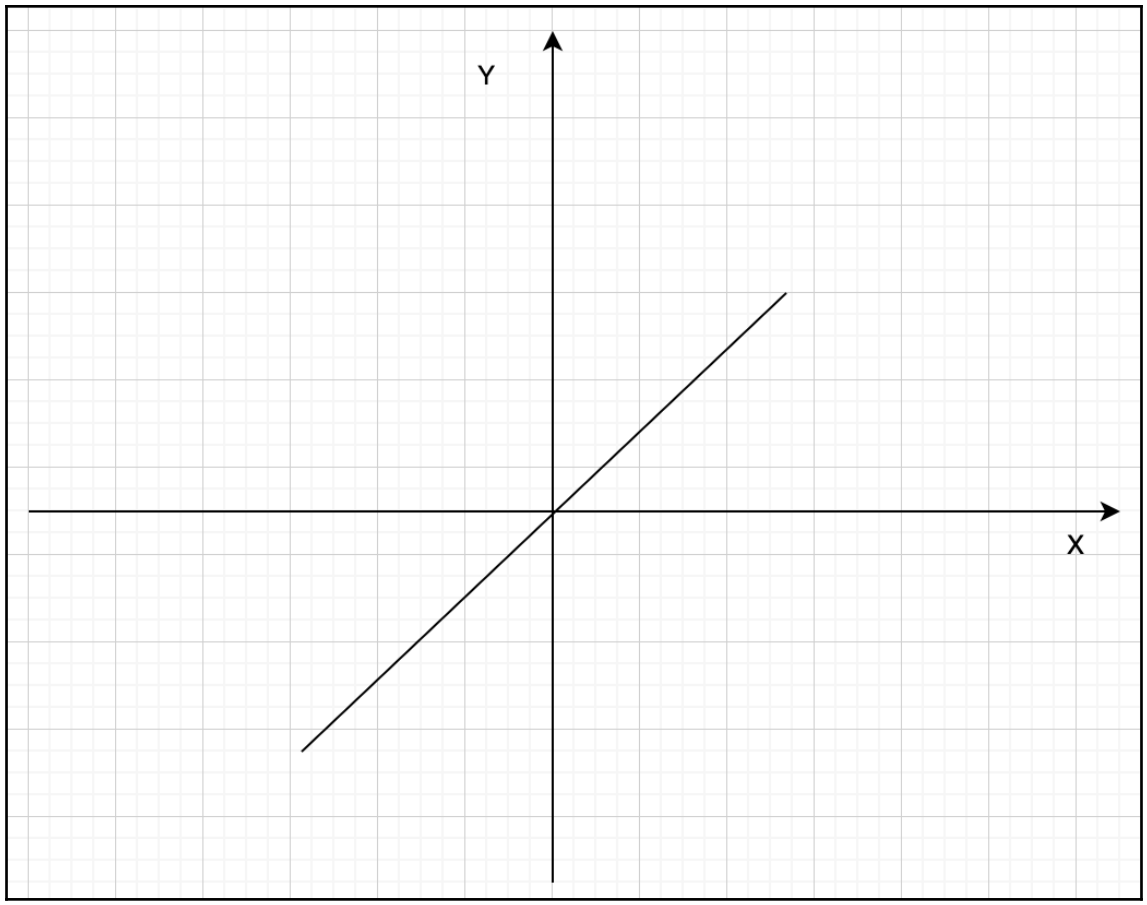

Known Data
New Response
It's an apple!
Known Response
These are apples
Model
New Data

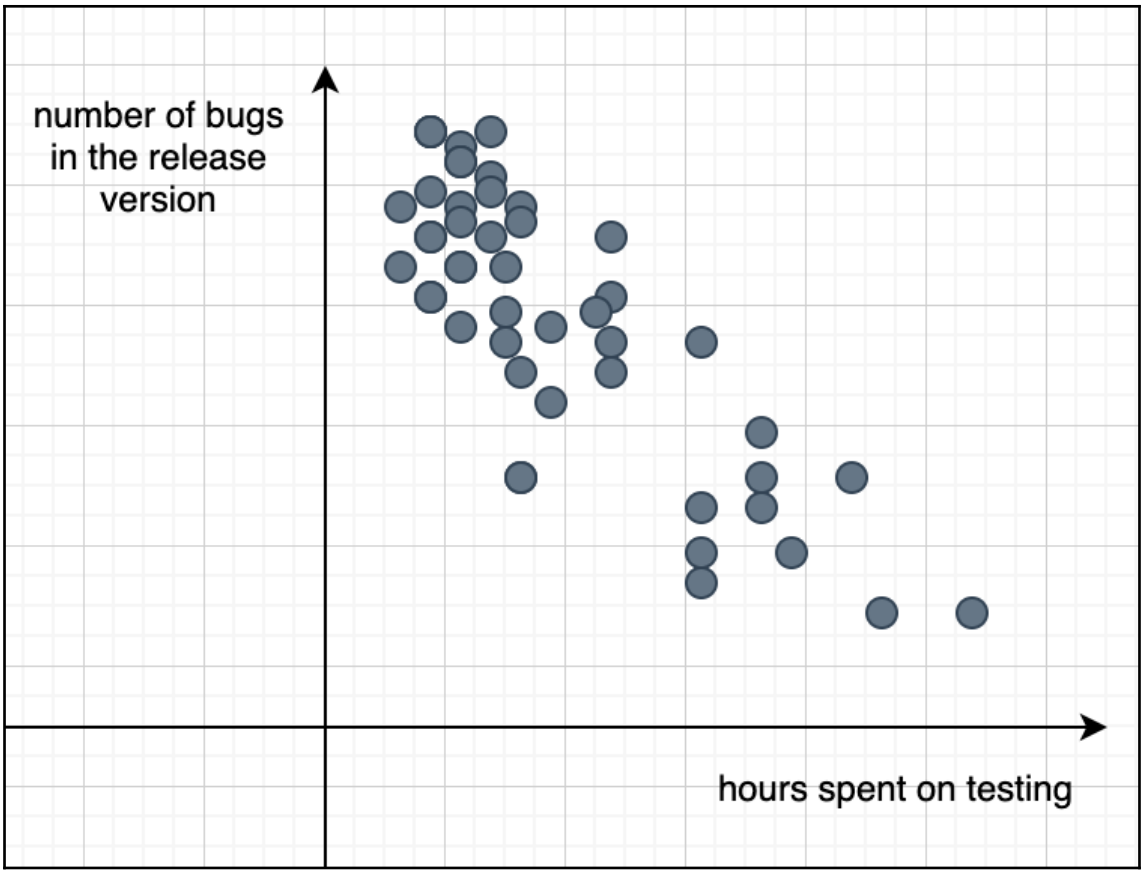

I can see
a pattern
Model
Input Data


Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic
reticulum
Mitochondrion
Dendrite
Dendritic branches

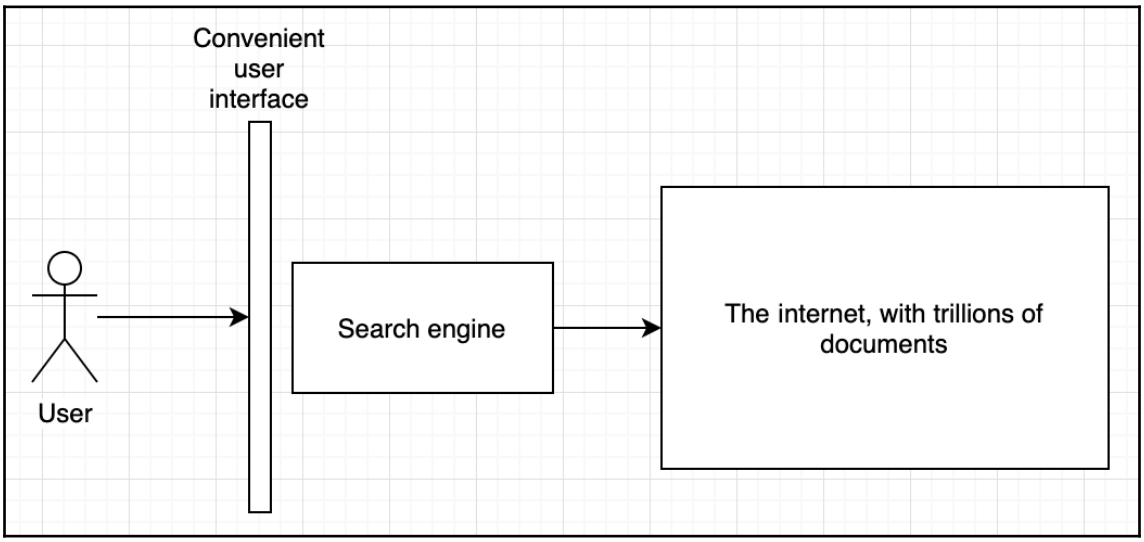

Hidden
Input
Output

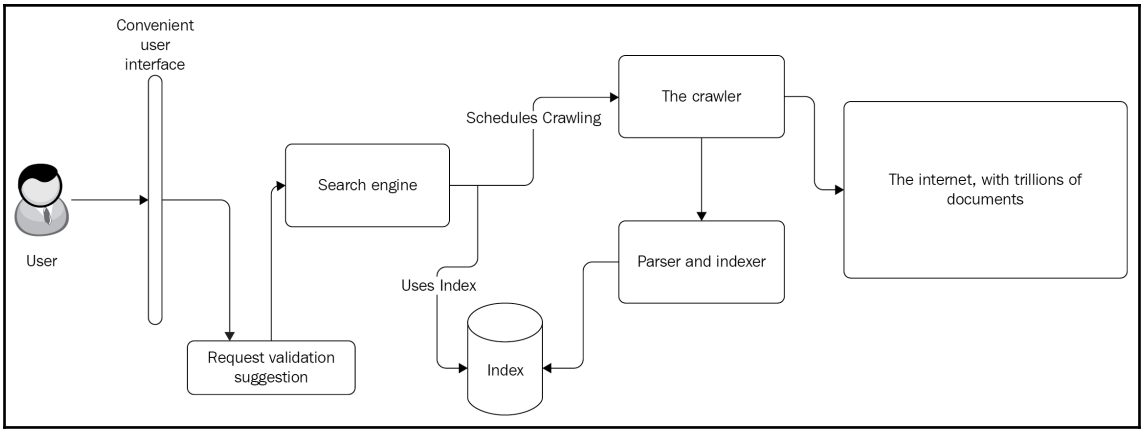



Y
X

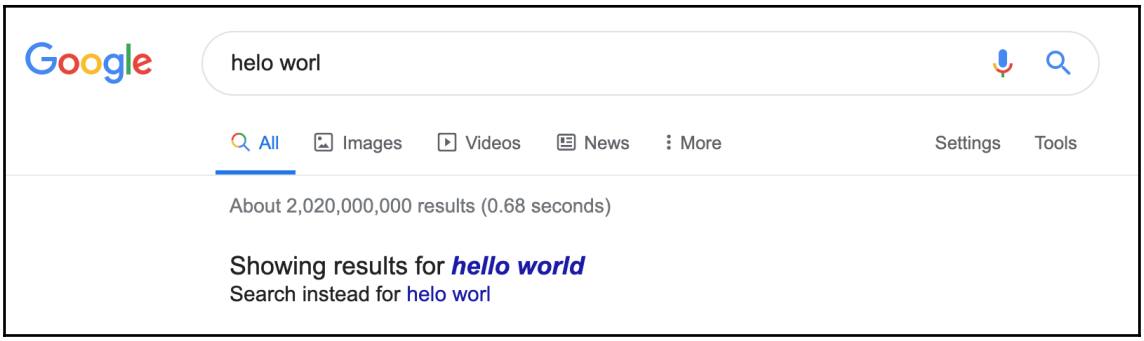

number of bugs
in the release
version
hours spent on testing

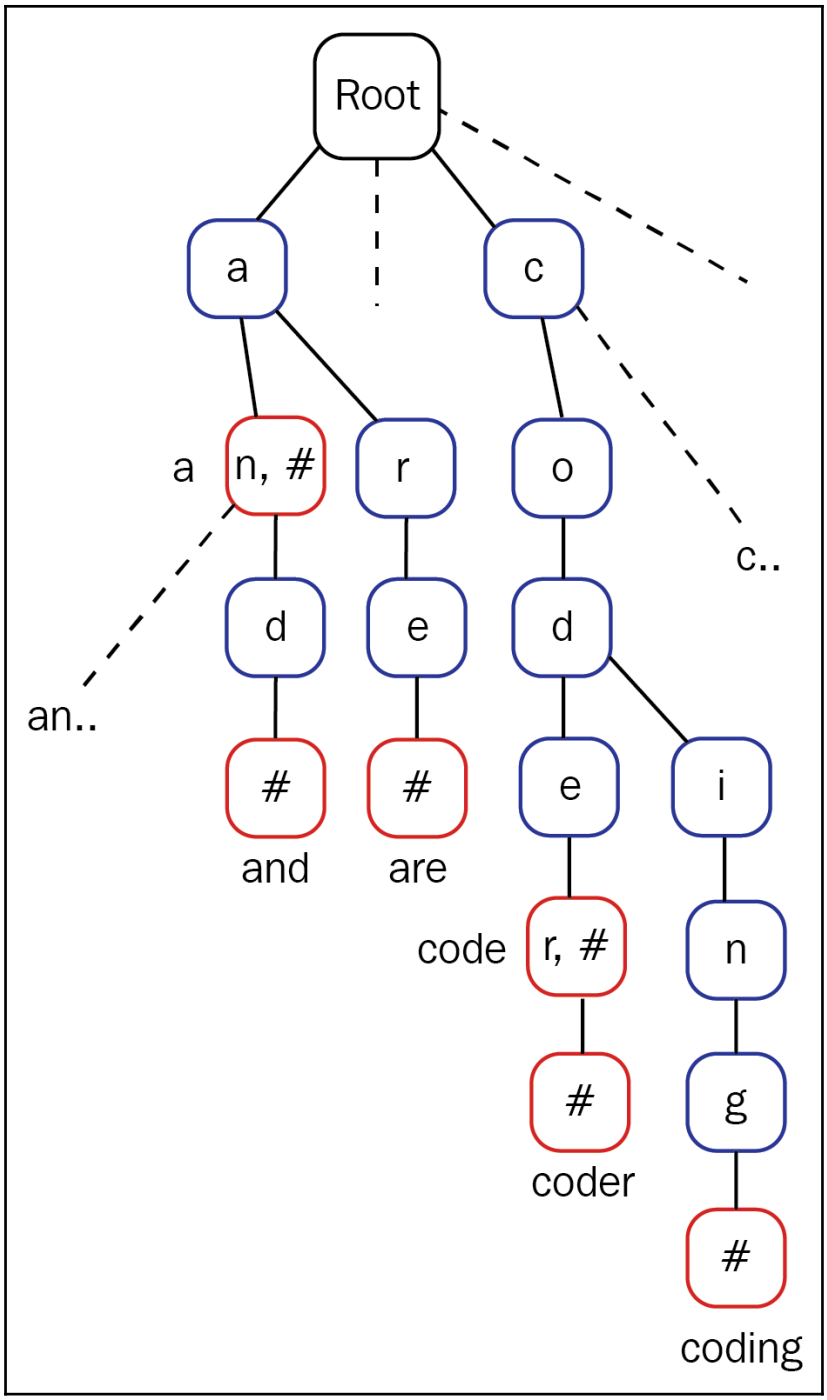
Chapter 16: Implementing a Dialog-Based Search Engine

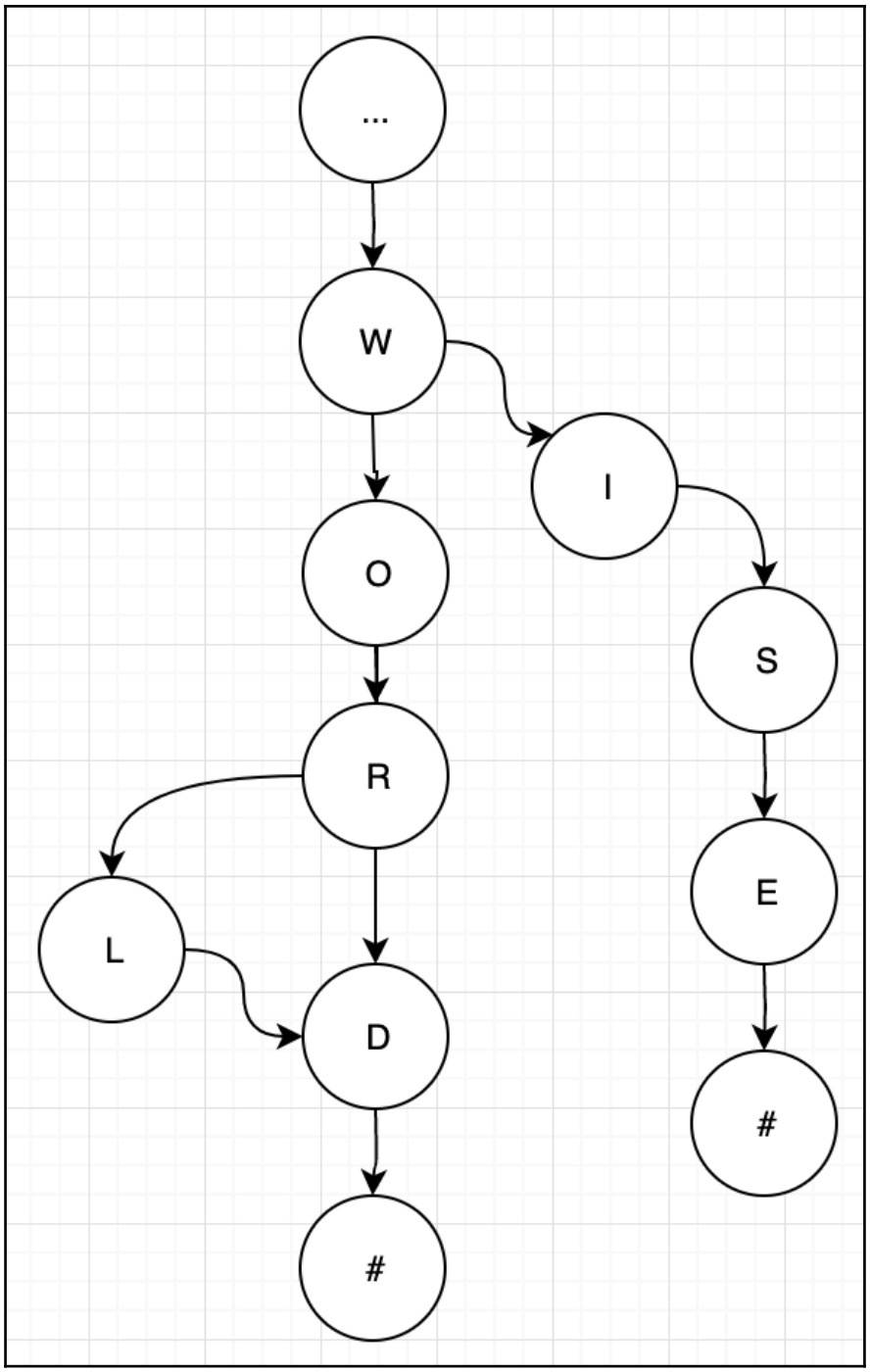


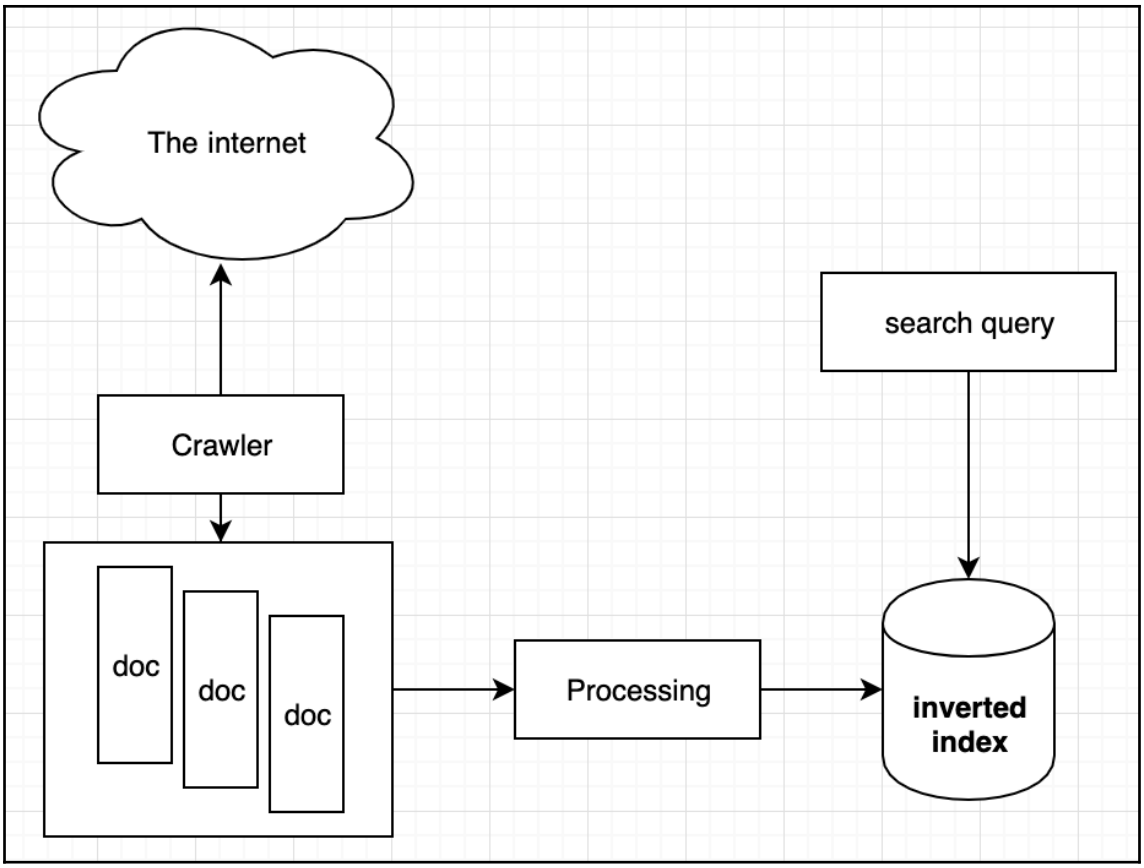



Google
mastering C++
mastering c++ programming pdf
mastering c++ pdf
mastering c++ programming
mastering c++ multithreading pdf
mastering c++ game development
mastering c++ multithreading
mastering c++ standard library features
mastering c++ 17 stl pdf
mastering c++ stl
mastering c++ programming book pdf
Google Search
I'm Feeling Lucky
Report inappropriate predictions


Google
helo worl
All
Images
Videos
News
More
Settings
Tools
About 2,020,000,000 results (0.68 seconds)
Showing results for hello world
Search instead for helo worl


Root
a
c
a
n, #
r
o
c..
an..
d
e
d
#
#
e
i
and
are
code
r, #
n
#
g
coder
#
coding


...
W
I
O
S
R
E
L
D
#
#


The internet
search query
Crawler
doc
doc
doc
Processing
inverted
index

Donald Knuth

American computer scientist

Donald Ervin Knuth is an American computer scientist, mathematician, and professor emeritus at Stanford University. He is the 1974 recipient of the ACM Turing Award, informally considered the Nobel Prize of computer science. He is the author of the multi-volume work The Art of Computer Programming. Wikipedia

Born: January 10, 1938 (age 82 years), Milwaukee, Wisconsin, United States

Spouse: Jill Knuth (m. 1961)

Education: California Institute of Technology (1960–1963), MORE

Awards: Turing Award, Kyoto Prize, MORE

People also search for View 10+ more

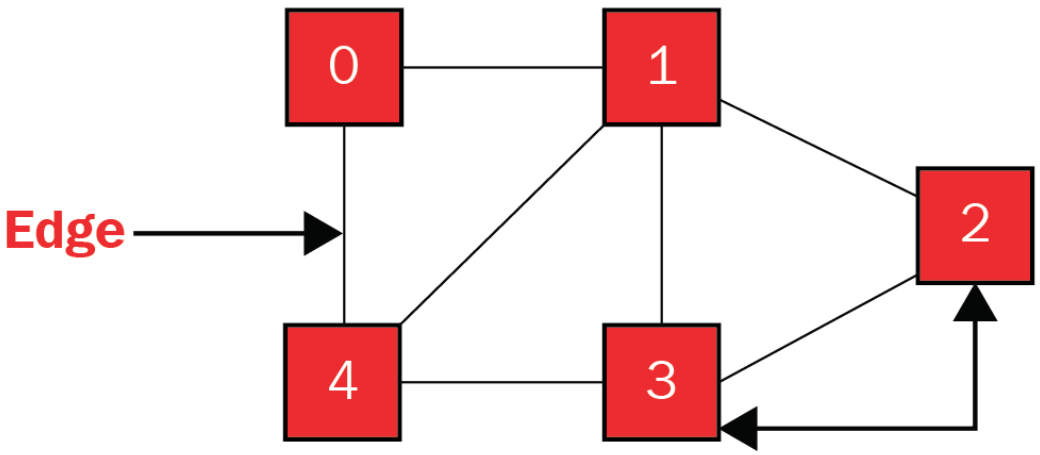

Robert Sedgewick

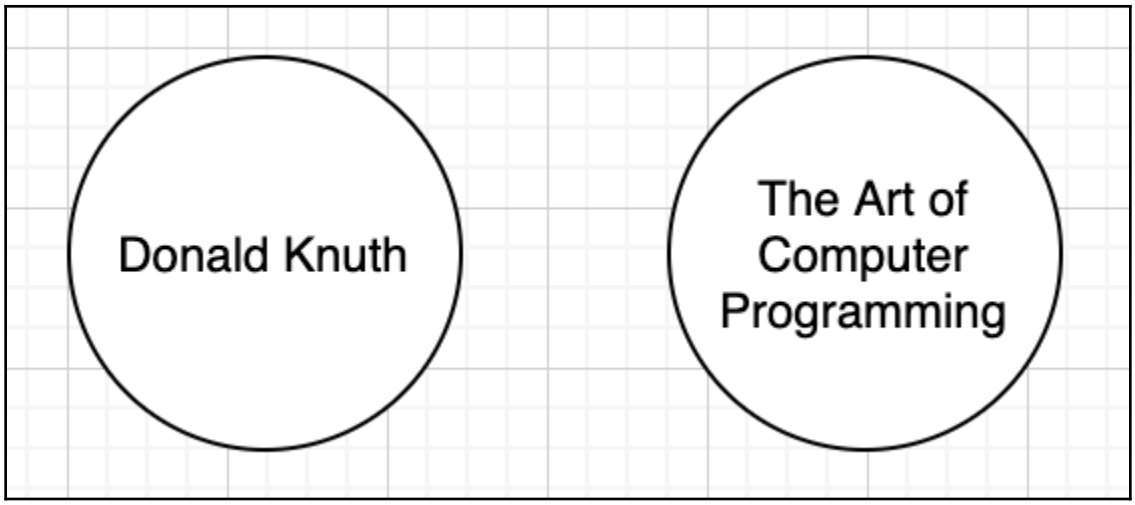

Alan Turing

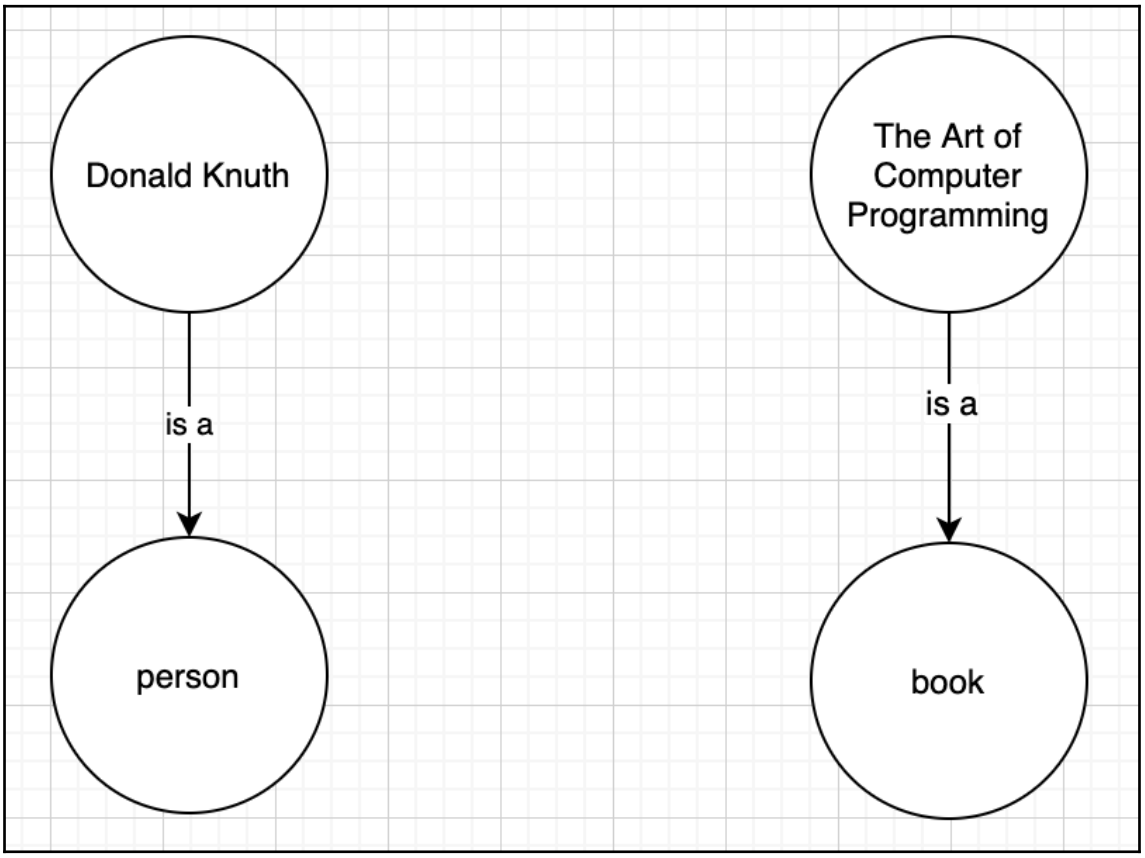

Edsger W. Dijkstra

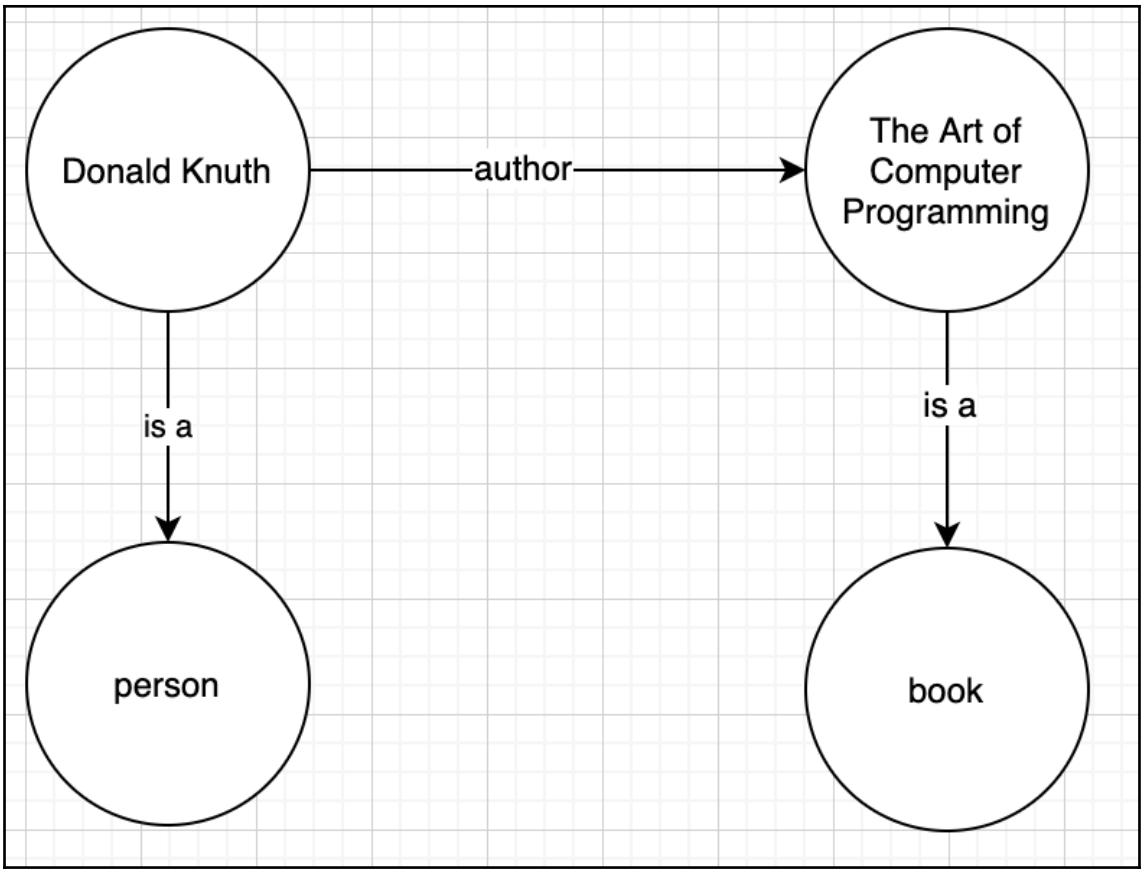

Vaughan Pratt

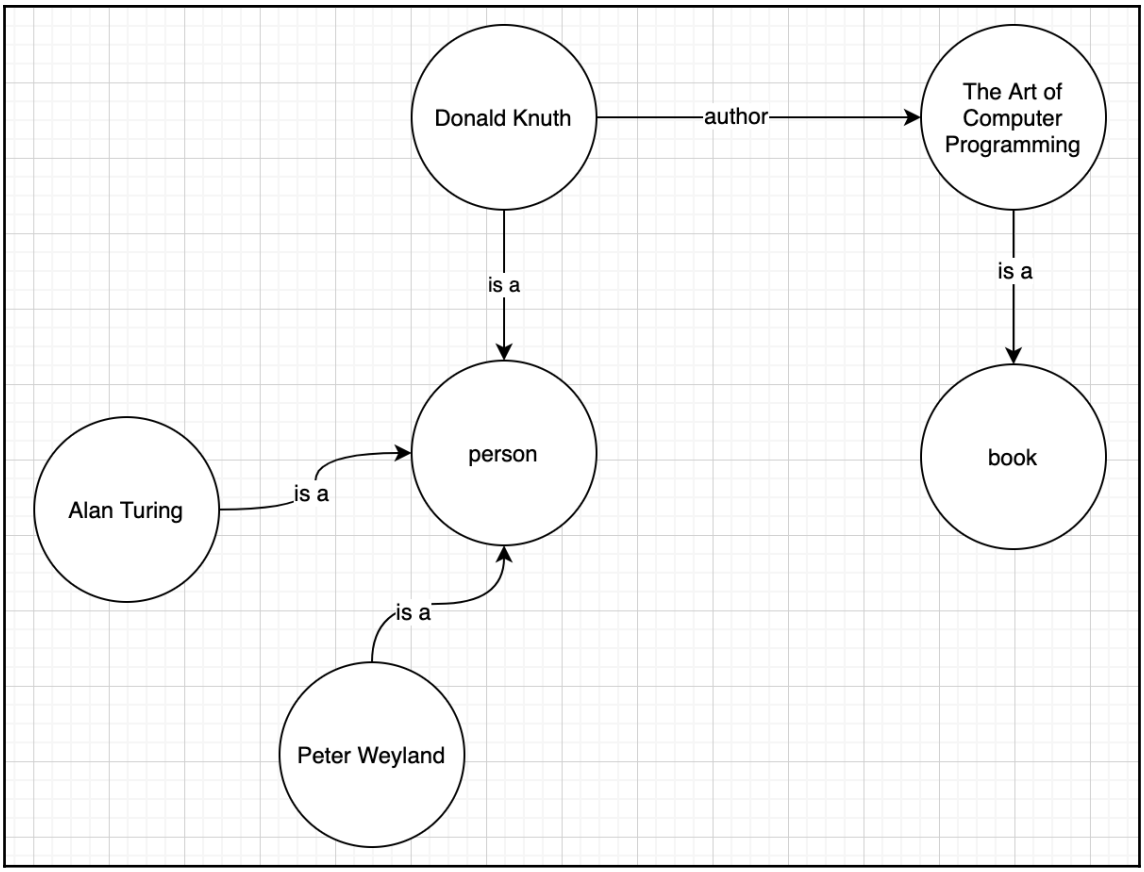

Ronald Graham

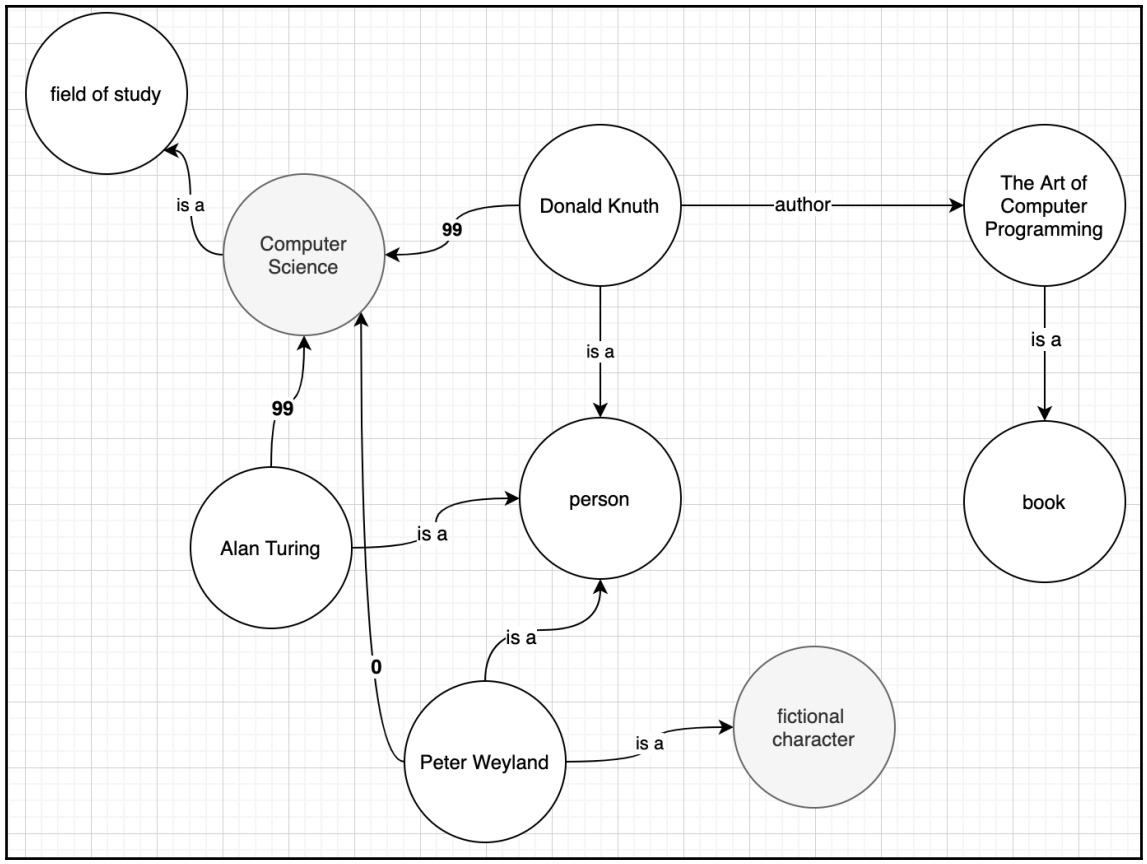


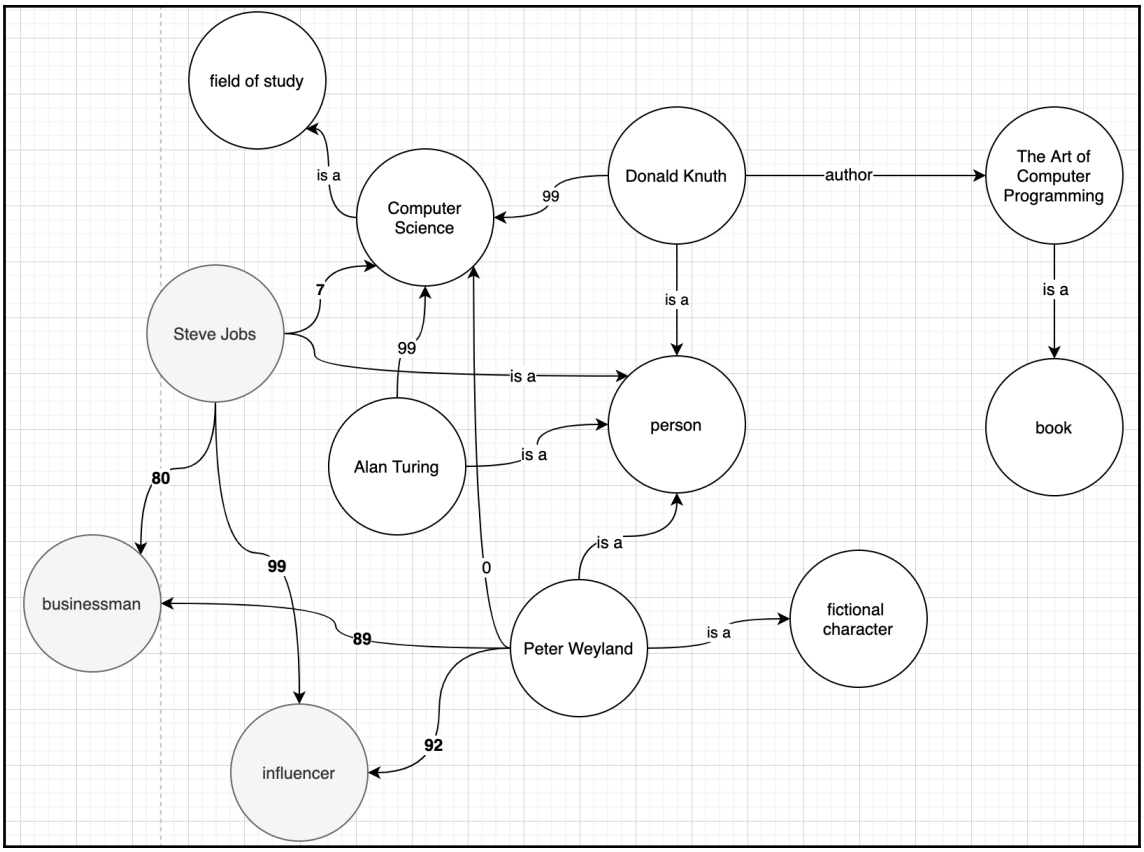

Donald Knuth
The Art of Computer Programming

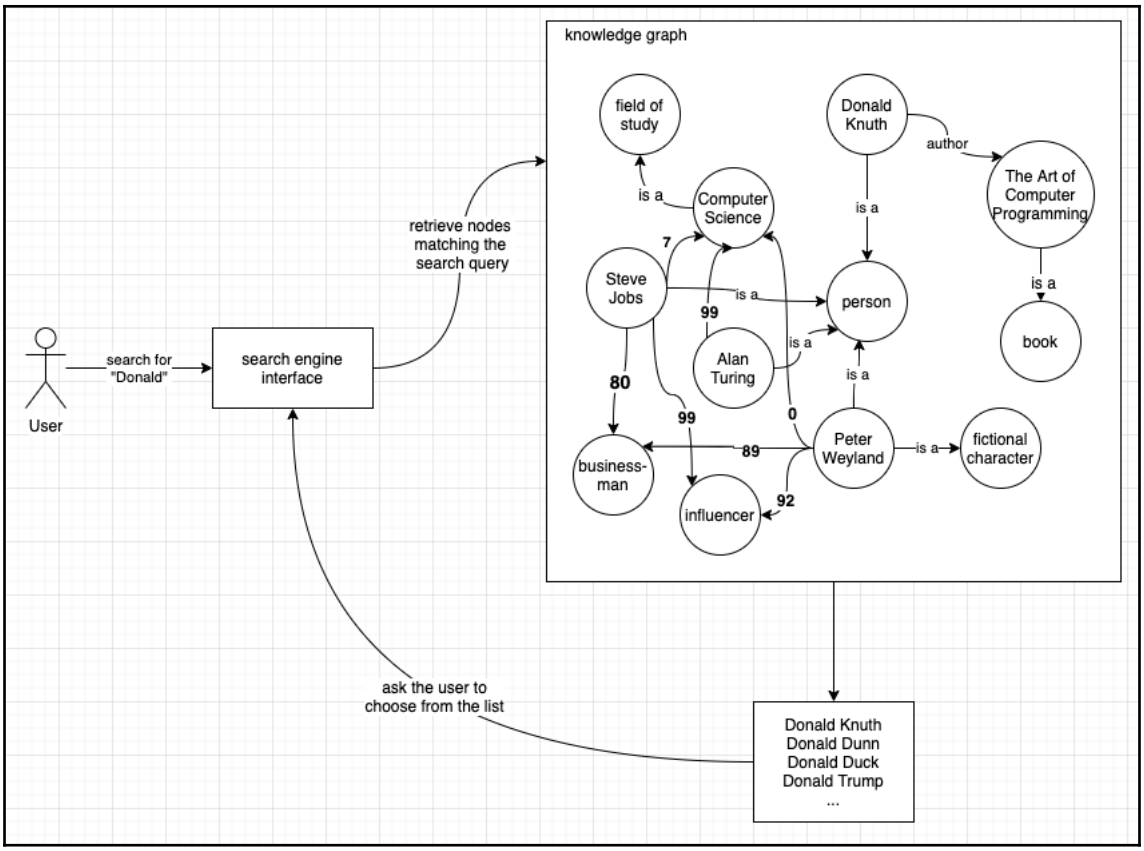

Donald Knuth
is a
person
The Art of Computer Programming
is a
book

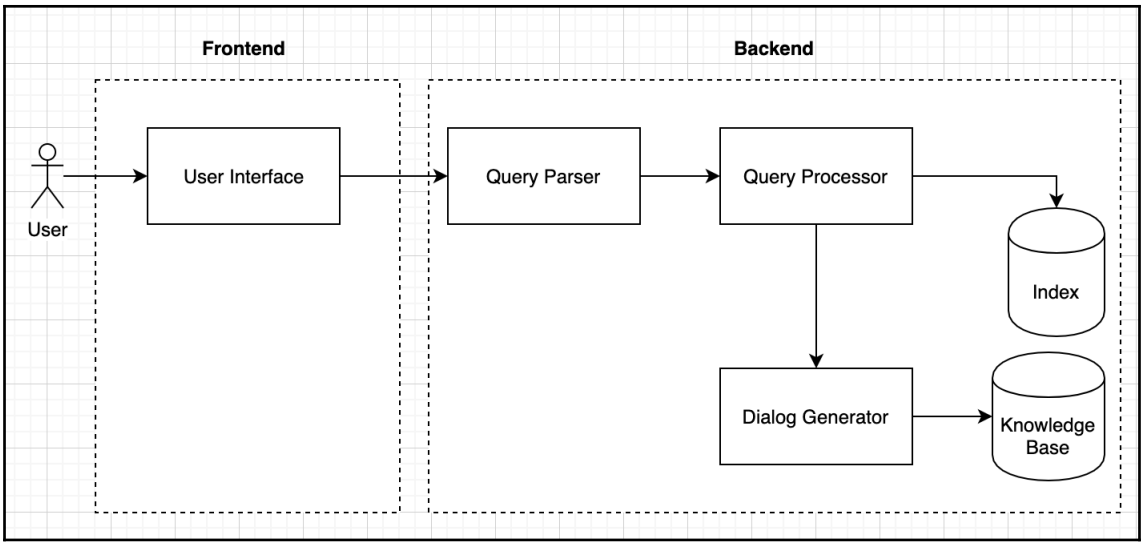

Donald Knuth
author
The Art of Computer Programming
is a
is a
person
book


Donald Knuth
author
The Art of Computer Programming
is a
is a
person
book
Alan Turing
is a
is a
Peter Weyland


field of study
is a
Computer Science
99
Donald Knuth
author
The Art of Computer Programming
is a
is a
99
person
book
Alan Turing
is a
is a
0
Peter Weyland
is a
fictional character


field of study
is a
Computer Science
99
Donald Knuth
author
The Art of Computer Programming
is a
is a
book
7
Steve Jobs
99
is a
person
Alan Turing
is a
80
99
0
is a
businessman
89
Peter Weyland
is a
fictional character
92
influencer


knowledge graph
field of study
Donald Knuth
author
The Art of Computer Programming
is a
Computer Science
is a
is a
book
7
Steve Jobs
is a
person
99
Alan Turing
is a
is a
80
99
0
Peter Weyland
is a
fictional character
89
business-man
influencer
92
retrieve nodes matching the search query
User
search for "Donald"
search engine interface
ask the user to choose from the list
Donald Knuth
Donald Dunn
Donald Duck
Donald Trump
...


Frontend
Backend
User
User Interface
Query Parser
Query Processor
Index
Dialog Generator
Knowledge Base