Thématique

Objects First with Java: A Practical Introduction Using BlueJ. By: David J. Barnes; Michael Kolling. Publisher: Pearson

Chapter 5 Functional Processing of Collections (Advanced)

- 5.1 An alternative look at themes from Chapter 4
- 5.2 Monitoring animal populations
- 5.3 A first look at lambdas
- 5.3.1 Processing a collection with a lambda
- 5.3.2 Basic syntax of a lambda
- 5.4 The forEach method of collections
- 5.4.1 Variations of lambda syntax
- 5.5 Streams
- 5.5.1 Filters, maps and reductions
- The filter function
- The map function
- The reduce function
- 5.5.2 Pipelines
- 5.5.3 The filter method
- 5.5.4 The map method
- 5.5.5 The reduce method
- 5.5.6 Removing from a collection
- 5.5.7 Other stream methods
- 5.6 Summary

Chapter 6 More-Sophisticated Behavior

- 6.1 Documentation for library classes
- 6.2 The TechSupport system
- 6.2.1 Exploring the TechSupport system
- 6.2.2 Reading the code
- 6.3 Reading class documentation
- 6.3.1 Interfaces versus implementation
- 6.3.2 Using library-class methods
- 6.3.3 Checking string equality
- 6.4 Adding random behavior
- 6.4.1 The Random class
- 6.4.2 Random numbers with limited range
- 6.4.3 Generating random responses
- 6.4.4 Reading documentation for parameterized classes
- 6.5 Packages and import
- 6.6 Using maps for associations
- 6.6.1 The concept of a map
- 6.6.2 Using a HashMap
- 6.6.3 Using a map for the TechSupport system

Progression du cours

Lundi 31 Janvier 2022 à 12:53
2 h
Cours Magistral n°1

Livre Chapitre 1 et Chapitre 2

https://www.bluej.org/

Chapter 1 Objects and Classes

- 1.1 Objects and classes
- 1.2 Creating objects
- 1.3 Calling methods
- 1.4 Parameters
- 1.5 Data types
- 1.6 Multiple instances
- 1.7 State
- 1.8 What is in an object?
- 1.9 Java code
- 1.10 Object interaction
- 1.11 Source code
- 1.12 Another example
- 1.13 Return values
- 1.14 Objects as parameters
- 1.15 Summary

Chapter 2 Understanding Class Definitions

- 2.1 Ticket machines
- 2.1.1 Exploring the behavior of a naïve ticket machine
- 2.2 Examining a class definition
- 2.3 The class header
- 2.3.1 Keywords
- 2.4 Fields, constructors, and methods
- 2.4.1 Fields
- 2.4.2 Constructors
- 2.5 Parameters: receiving data
- 2.5.1 Choosing variable names
- 2.6 Assignment
- 2.7 Methods
- 2.8 Accessor and mutator methods
- 2.9 Printing from methods
- 2.10 Method summary
- 2.11 Summary of the naíve ticket machine
- 2.12 Reflecting on the design of the ticket machine
- 2.13 Making choices: the conditional statement
- 2.14 A further conditional-statement example
- 2.15 Scope highlighting
- 2.16 Local variables
- 2.17 Fields, parameters, and local variables

- 6.7 Using sets
- 6.8 Dividing strings
- 6.9 Finishing the TechSupport system
- 6.10 Autoboxing and wrapper classes
- 6.10.1 Maintaining usage counts
- 6.11 Writing class documentation
- 6.11.1 Using javadoc in BlueJ
- 6.11.2 Elements of class documentation
- 6.12 Public versus private
- 6.12.1 Information hiding
- 6.12.2 Private methods and public fields
- 6.13 Learning about classes from their interfaces
- 6.13.1 The scribble demo
- 6.13.2 Code completion
- 6.13.3 The bouncing-balls demo
- 6.14 Class variables and constants
- 6.14.1 The static keyword
- 6.14.2 Constants
- 6.15 Class methods
- 6.15.1 Static methods
- 6.15.2 Limitations of class methods
- 6.16 Executing without BlueJ
- 6.16.1 The main method
- 6.17 Further advanced material
- 6.17.1 Polymorphic collection types (Advanced)
- 6.17.2 The collect method of streams (Advanced)
- 6.17.3 Method references (Advanced)
- 6.18 Summary

Chapter 7 Fixed-Size Collections—Arrays

- 7.1 Fixed-size collections
- 7.2 Arrays
- 7.3 A log-file analyzer
- 7.3.1 Declaring array variables
- 7.3.2 Creating array objects
- 7.3.3 Using array objects
- 7.3.4 Analyzing the log file
- 7.4 The for loop
- 7.4.1 Arrays and the for-each loop
- 7.4.2 The for loop and iterators
- 7.5 The automaton project
- 7.5.1 The conditional operator
- 7.5.2 First and last iterations
- 7.5.3 Lookup tables
- 7.6 Arrays of more than one dimension (advanced)
- 7.6.1 The brain project
- 7.6.2 Setting up the array
- 7.7 Arrays and streams (advanced)
- 7.8 Summary

Chapter8 Designing Classes

- 8.1 Introduction
- 8.2 The world-of-zuul game example
- 8.3 Introduction to coupling and cohesion
- 2.18 Summary of the better ticket machine
- 2.19 Self-review exercises
- 2.20 Reviewing a familiar example
- 2.21 Calling methods
- 2.22 Experimenting with expressions: the Code Pad
- 2.23 Summary

Ressources : https://www.bluej.org/objects-first/index.html

Lundi 31 Janvier 2022 à 13:01
2 h
Cours magistral n°2

Livre Chapitre 3 et chapitre 4

Chapter 3 Object Interaction

- 3.1 The clock example
- 3.2 Abstraction and modularization
- 3.3 Abstraction in software
- 3.4 Modularization in the clock example
- 3.5 Implementing the clock display
- 3.6 Class diagrams versus object diagrams
- 3.7 Primitive types and object types
- 3.8 The NumberDisplay class
- 3.8.1 The logical operators
- 3.8.2 String concatenation
- 3.8.3 The modulo operator
- 3.9 The ClockDisplay class
- 3.10 Objects creating objects
- 3.11 Multiple constructors
- 3.12 Method calls
- 3.12.1 Internal method calls
- 3.12.2 External method calls
- 3.12.3 Summary of the clock display
- 3.13 Another example of object interaction
- 3.13.1 The mail-system example
- 3.13.2 The this keyword
- 3.14 Using a debugger
- 3.14.1 Setting breakpoints
- 3.14.2 Single stepping
- 3.14.3 Stepping into methods
- 3.15 Method calling revisited
- 3.16 Summary

Chapter 4 Grouping Objects

- 4.1 Building on themes from Chapter 3
- 4.2 The collection abstraction
- 4.3 An organizer for music files
- 4.4 Using a library class
- 4.4.1 Importing a library class
- 4.4.2 Diamond notation
- 4.4.3 Key methods of ArrayList
- 4.5 Object structures with collections
- 4.6 Generic classes

- 8.4 Code duplication
- 8.5 Making extensions
- 8.5.1 The task
- 8.5.2 Finding the relevant source code
- 8.6 Coupling
- 8.6.1 Using encapsulation to reduce coupling
- 8.7 Responsibility-driven design
- 8.7.1 Responsibilities and coupling
- 8.8 Localizing change
- 8.9 Implicit coupling
- 8.10 Thinking ahead
- 8.11 Cohesion
- 8.11.1 Cohesion of methods
- 8.11.2 Cohesion of classes
- 8.11.3 Cohesion for readability
- 8.11.4 Cohesion for reuse
- 8.12 Refactoring
- 8.12.1 Refactoring and testing
- 8.12.2 An example of refactoring
- 8.13 Refactoring for language independence
- 8.13.1 Enumerated types
- 8.13.2 Further decoupling of the command interface
- 8.14 Design guidelines
- 8.15 Summary

Chapter 9 Well-Behaved Objects

- 9.1 Introduction
- 9.2 Testing and debugging
- 9.3 Unit testing within BlueJ
- 9.3.1 Using inspectors
- 9.3.2 Positive versus negative testing
- 9.4 Test automation
- 9.4.1 Regression testing
- 9.4.2 Automated testing using JUnit
- 9.4.3 Recording a test
- 9.4.4 Fixtures
- 9.5 Refactoring to use streams (advanced)
- 9.6 Debugging
- 9.7 Commenting and style
- 9.8 Manual walkthroughs
- 9.8.1 A high-level walkthrough
- 9.8.2 Checking state with a walkthrough
- 9.8.3 Verbal walkthroughs
- 9.9 Print statements
- 9.9.1 Turning debugging information on or off
- 9.10 Debuggers
- 9.11 Debugging streams (advanced)
- 9.12 Choosing a debugging strategy
- 9.13 Putting the techniques into practice
- 9.14 Summary

Part 2 Application Structures

Chapter 10 Improving Structure with Inheritance

- 10.1 The network example
- 4.7 Numbering within collections
- 4.7.1 The effect of removal on numbering
- 4.7.2 The general utility of numbering with collections
- 4.8 Playing the music files
- 4.8.1 Summary of the music organizer
- 4.9 Processing a whole collection
- 4.9.1 The for-each loop
- 4.9.2 Selective processing of a collection
- 4.9.3 A limitation of using strings
- 4.9.4 Summary of the for-each loop
- 4.10 Indefinite iteration
- 4.10.1 The while loop
- 4.10.2 Iterating with an index variable
- 4.10.3 Searching a collection
- 4.10.4 Some non-collection examples
- 4.11 Improving structure—the Track class
- 4.12 The Iterator type
- 4.12.1 Index access versus iterators
- 4.12.2 Removing elements
- 4.13 Summary of the music-organizer project
- 4.14 Another example: an auction system
- 4.14.1 Getting started with the project
- 4.14.2 The null keyword
- 4.14.3 The Lot class
- 4.14.4 The Auction class
- 4.14.5 Anonymous objects
- 4.14.6 Chaining method calls
- 4.14.7 Using collections
- 4.15 Summary

- 10.1.1 The network project: classes and objects
- 10.1.2 Network source code
- 10.1.3 Discussion of the network application
- 10.2 Using inheritance
- 10.3 Inheritance hierarchies
- 10.4 Inheritance in Java
- 10.4.1 Inheritance and access rights
- 10.4.2 Inheritance and initialization
- 10.5 Network: adding other post types
- 10.6 Advantages of inheritance (so far)
- 10.7 Subtyping
- 10.7.1 Subclasses and subtypes
- 10.7.2 Subtyping and assignment
- 10.7.3 Subtyping and parameter passing
- 10.7.4 Polymorphic variables
- 10.7.5 Casting
- 10.8 The Object class
- 10.9 The collection hierarchy
- 10.10 Summary

Chapter 11 More about Inheritance

- 11.1 The problem: network's display method
- 11.2 Static type and dynamic type
- 11.2.1 Calling display from NewsFeed
- 11.3 Overriding
- 11.4 Dynamic method lookup
- 11.5 super call in methods
- 11.6 Method polymorphism
- 11.7 Object methods: toString
- 11.8 Object equality: equals and hashCode
- 11.9 Protected access
- 11.10 The instanceof operator
- 11.11 Another example of inheritance with overriding
- 11.12 Summary

Handling Errors