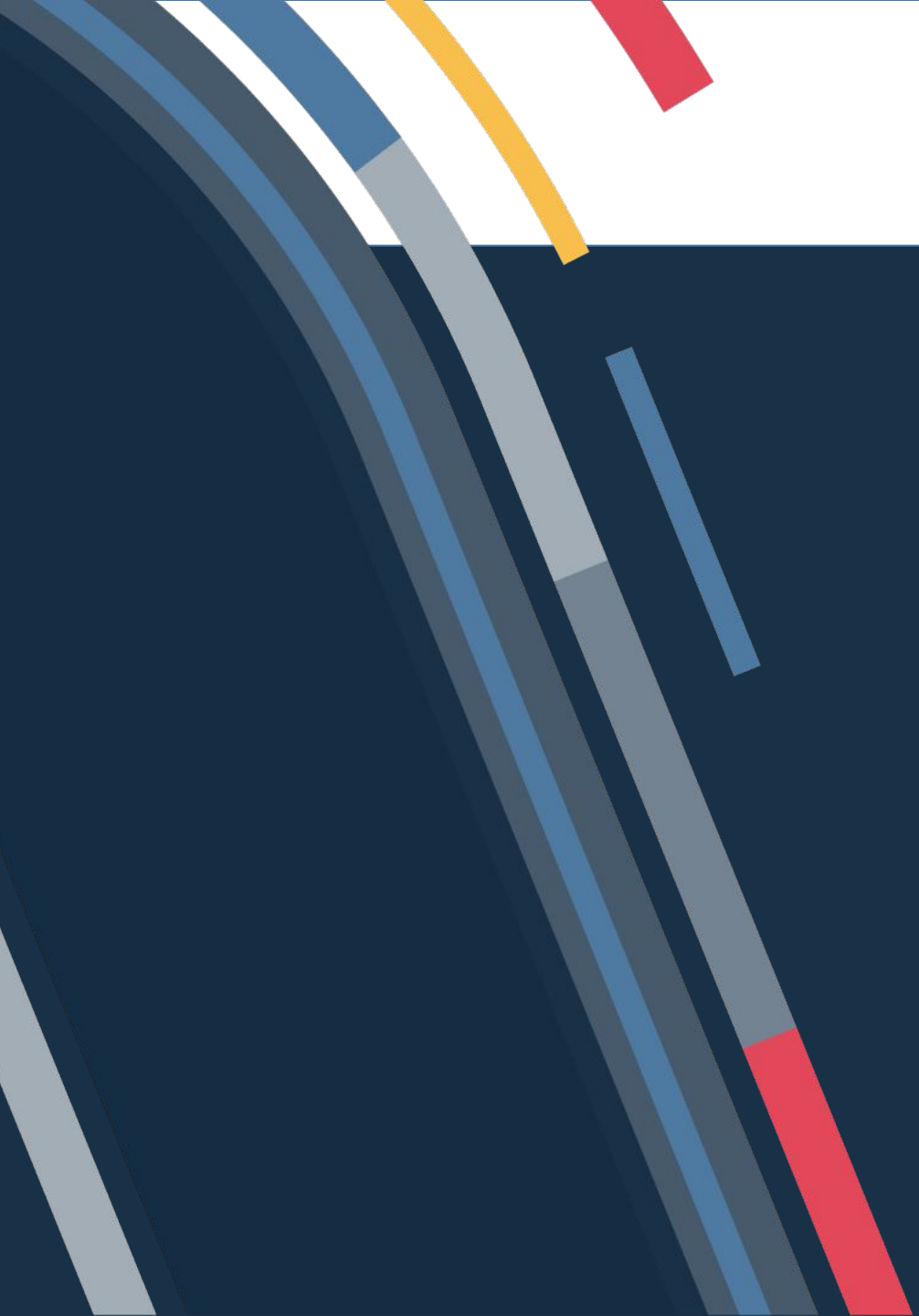


Netcracker

Java Introduction Object Oriented Programming

Gleb Akimov, Boris Larenyshev

Java Course Overview

- An overview of Java, Object oriented programming paradigm
- Programming Structures in Java (comments, operators, data types, variables, strings, arrays)
- Objects and classes
- Inheritance
- Packages and interfaces
- Collections
- Inner classes
- Generic programming
- String handling
- Exceptions, Assertions
- Enumerations, Autoboxing, Annotations

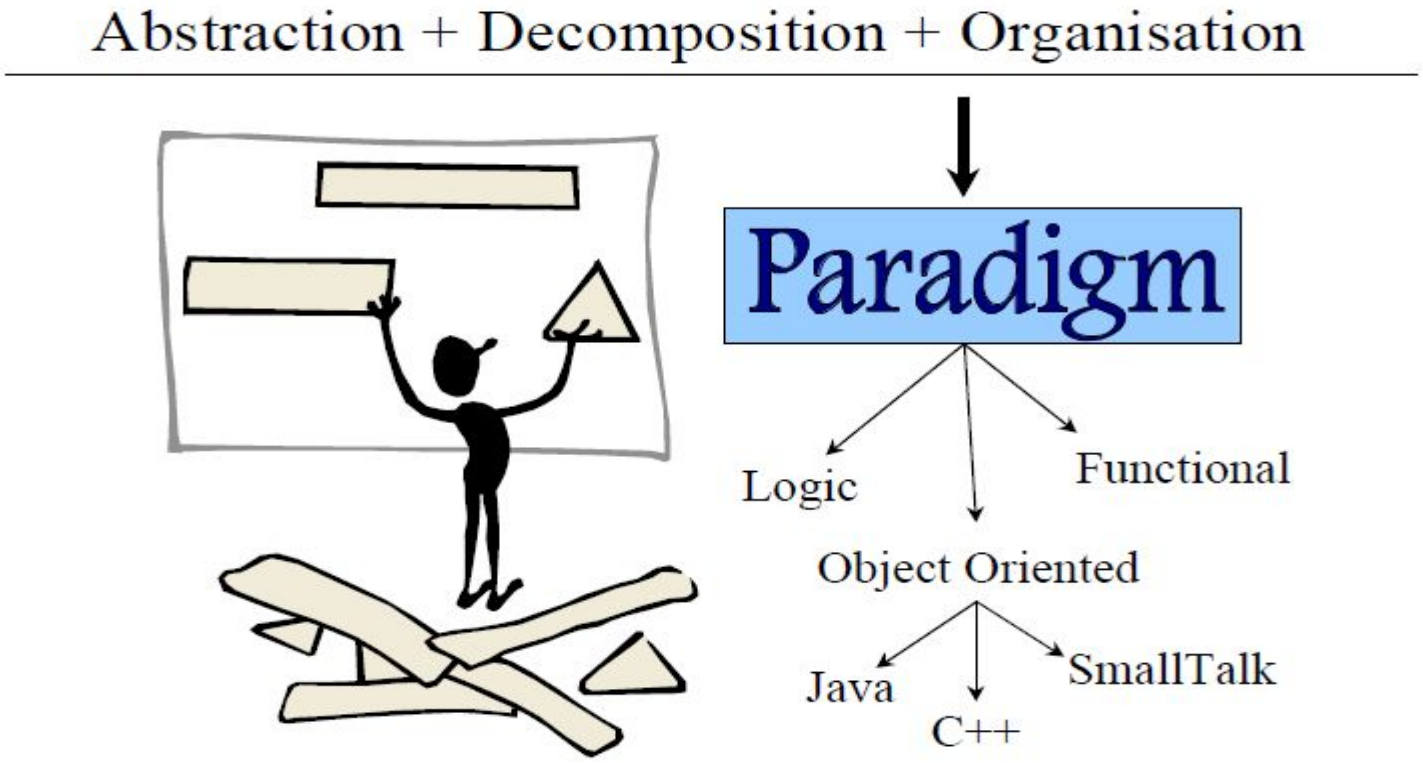
© NetCracker 2017

Java Books and Internet resources

1. Java The Complete Reference (9th edition), Herbert Shildt
2. Core Java, Volume I – Fundamentals (10th edition), Cay S. Horstmann
3. Thinking in Java (4th edition), Bruce Eckel
4. Effective Java (2th edition), Joshua Bloch
5. GoF's Desighn Pattrens in Java
6. Code Complete, Steve McConnell
7. http://www.oracle.com/technetwork/java/index.html
8. https://www.tutorialspoint.com/
9. http://it-ebooks.info/
10. http://stackoverflow.com/

© NetCracker 2017

Basic Concepts in Object Oriented Programming



We use the word paradigm to mean “any example or model”.
This usage of the word was popularised by the science historian Thomas Kuhn.
He used the term to describe a set of theories, standards and methods that together represent a way of organising knowledge—a way of viewing the world.

© NetCracker 2017

Functional/Procedural Paradigm

We think in terms of **functions** acting on **data**

ABSTRACTION: Think of the problem in terms of a process that solves it.

DECOMPOSITION: Break your processing down into smaller manageable processing units (functions).

ORGANIZATION: Set up your functions so that they call each other (function calls, arguments, etc.)

FIRST: define your set of data structures (types, etc.)

THEN: define your set of functions acting upon the data structures.

© NetCracker 2017

Object Oriented Paradigm

We think in terms of objects interacting:

ABSTRACTION: Think in terms of independent agents (objects) working together.

DECOMPOSITION: Define the kinds of objects on which to split the global task.

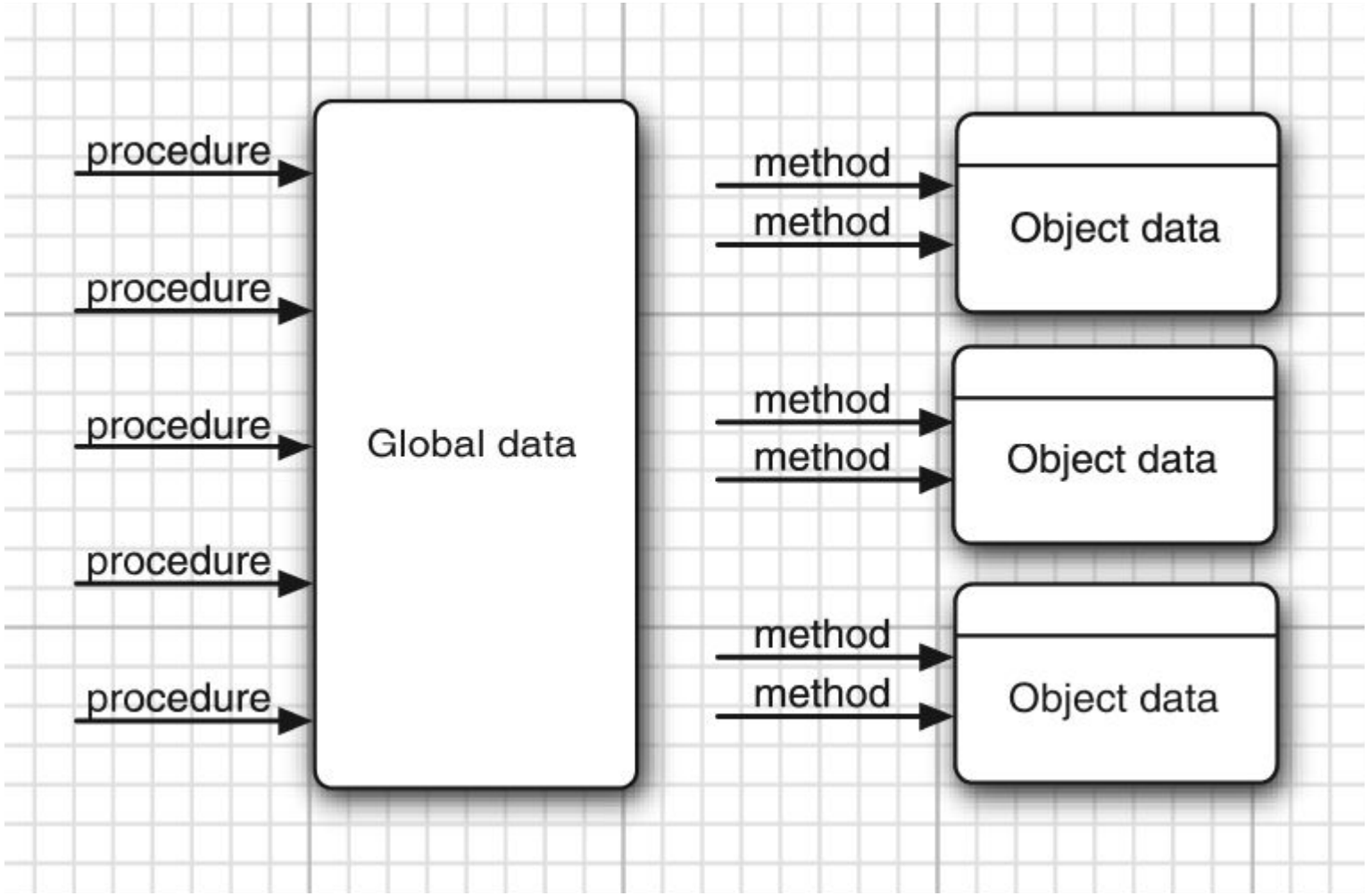
ORGANIZATION: Create the appropriate number of objects of each kind.

FIRST: Define the behavior and properties of objects of the different kinds we have defined.

THEN: Set up objects of each kind and put them to work.

© NetCracker 2017

Procedural vs. OO programming



© NetCracker 2017

Alan Kay summarized five basic characteristics of **Smalltalk**, the **first successful object-oriented language and one of the languages upon which Java is based**. These characteristics represent a pure approach to object-oriented programming:

- **Everything is an object**. Think of an object as a fancy variable; it stores data, but you can "make requests" to that object, asking it to perform operations on itself. In theory, you can take any conceptual component in the problem you're trying to solve (dogs, buildings, services, etc.) and represent it as an object in your program.
- **A program is a bunch of objects telling each other what to do by sending messages**. To make a request of an object, you "send a message" to that object. More concretely, you can think of a message as a request to call a method that belongs to a particular object.
- **Each object has its own memory made up of other objects**. Put another way, you create a new kind of object by making a package containing existing objects. Thus, you can build complexity into a program while hiding it behind the simplicity of objects.
- **Every object has a type**. Using the parlance, each object is an *instance* of a *class*, in which "class" is synonymous with "type." The most important distinguishing characteristic of a class is "What messages can you send to it?"
- **All objects of a particular type can receive the same messages**. This is actually a loaded statement, as you will see later. Because an object of type "circle" is also an object of type "shape," a circle is guaranteed to accept shape messages.

© NetCracker 2017

The Three OOP Principles

Encapsulation

Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.

Inheritance

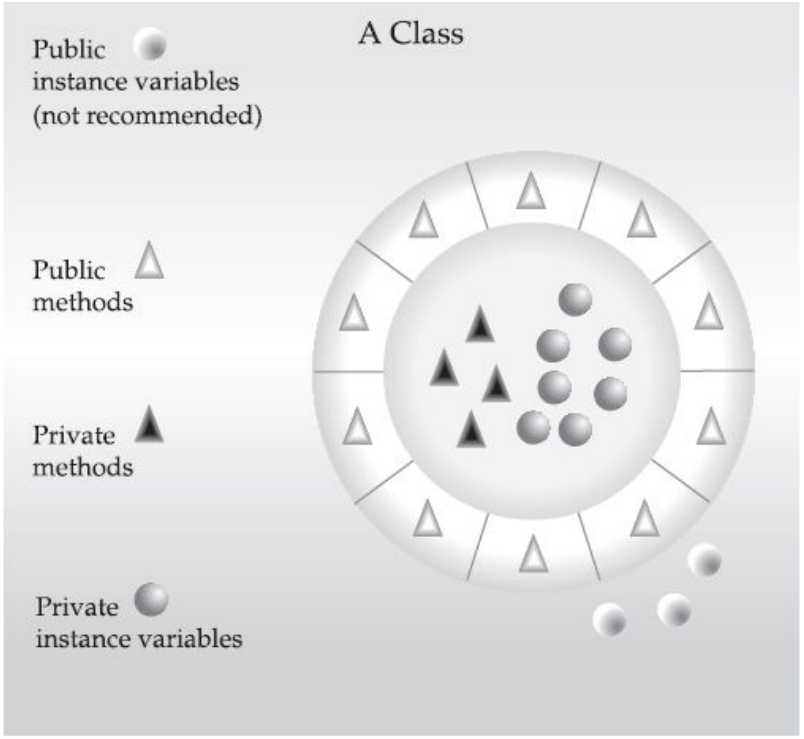
Inheritance is the process by which one object acquires the properties of another object. Thisis important because it supports the concept of hierarchical classification.

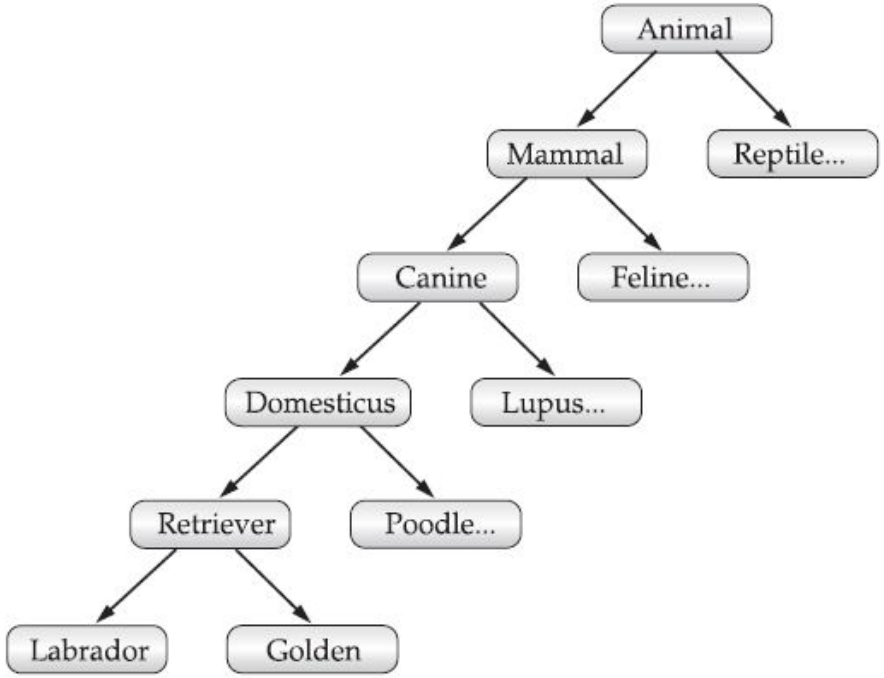
Polymorphism

Polymorphism (from Greek, meaning “many forms”) is a feature that allows one interface to be used for a general class of actions.

© NetCracker 2017

Encapsulation and Inheritance examples





© NetCracker 2017

Objects and Classes

Objects

In **object-oriented** programming we create **software objects** that model **real world objects**.

Software objects are modeled after real-world objects in that they too have **state** and **behavior**.

A **software object** maintains its state in one or more variables.

A **variable** is an item of data named by an identifier. **A software object implements its behavior with methods**.

A **method** is a function associated with an object.

© NetCracker 2017

Classes

In object-oriented software, it's possible to have **many objects of the same kind that share characteristics**: rectangles, employee records, video clips, and so on.

A class is a software blueprint for objects.

A class is used to manufacture or create objects.

© NetCracker 2017

Messages

Software objects interact and communicate with each other by sending messages to each other.

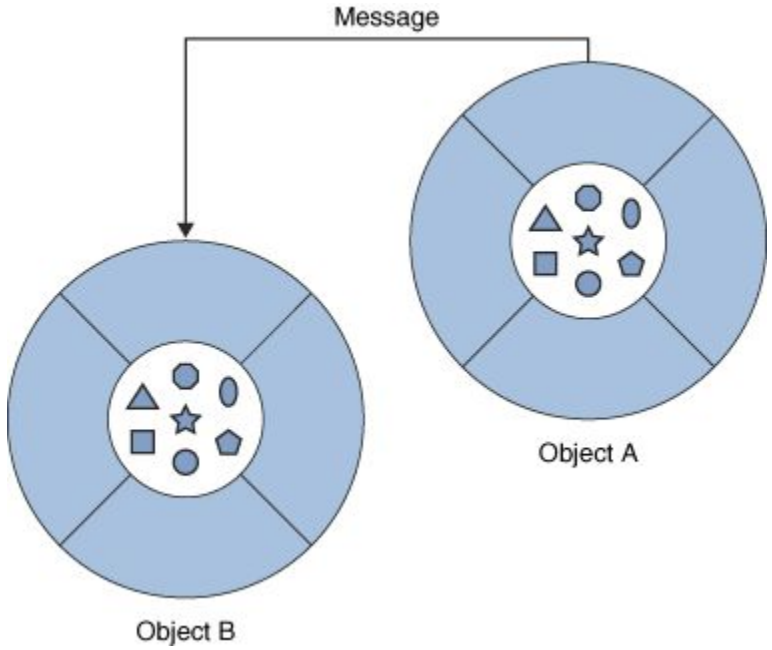
When object A wants object B to perform one of B's methods, object A

sends a message to object B

There are three parts of a message: **The three parts for the message**

System.out.println{"Hello World"}; are:

- The object to which the message is addressed (System.out)
- The name of the method to perform (println)
- Any parameters needed by the method ("Hello World!")



© NetCracker 2017

Evolution of the Java Language

Version	Year	New Language Features	Number of Classes and Interfaces
1.0	1996	The language itself	211
1.1	1997	Inner classes	477
1.2	1998	The `strictfp` modifier	1,524
1.3	2000	None	1,840
1.4	2002	Assertions	2,723
5.0	2004	Generic classes, "for each" loop, varargs, autoboxing, metadata, enumerations, static import	3,279
6	2006	None	3,793
7	2011	Switch with strings, diamond operator, binary literals, exception handling enhancements	4,024
8	2014	Lambda expressions, interfaces with default methods, stream and date/time libraries	4,240

© NetCracker 2017

The Java Programming Environment

Java Jargon

Name	Acronym	Explanation
Java Development Kit	JDK	The software for programmers who want to write Java programs
Java Runtime Environment	JRE	The software for consumers who want to run Java programs
Server JRE	—	The software for running Java programs on servers
Standard Edition	SE	The Java platform for use on desktops and simple server applications
Enterprise Edition	EE	The Java platform for complex server applications
Micro Edition	ME	The Java platform for use on cell phones and other small devices
Java FX	—	An alternate toolkit for graphical user interfaces that is included in Oracle's Java SE distribution
OpenJDK	—	A free and open source implementation of Java SE. It does not include browser integration or JavaFX.
Java 2	J2	An outdated term that described Java versions from 1998 until 2006
Software Development Kit	SDK	An outdated term that described the JDK from 1998 until 2006
Update	u	Oracle's term for a bug fix release
NetBeans	—	Oracle's integrated development environment

© NetCracker 2017

Q&A

Thank You