Textbook: Object-Oriented Analysis & Design

Chapter 2: Part-A
The Object Model

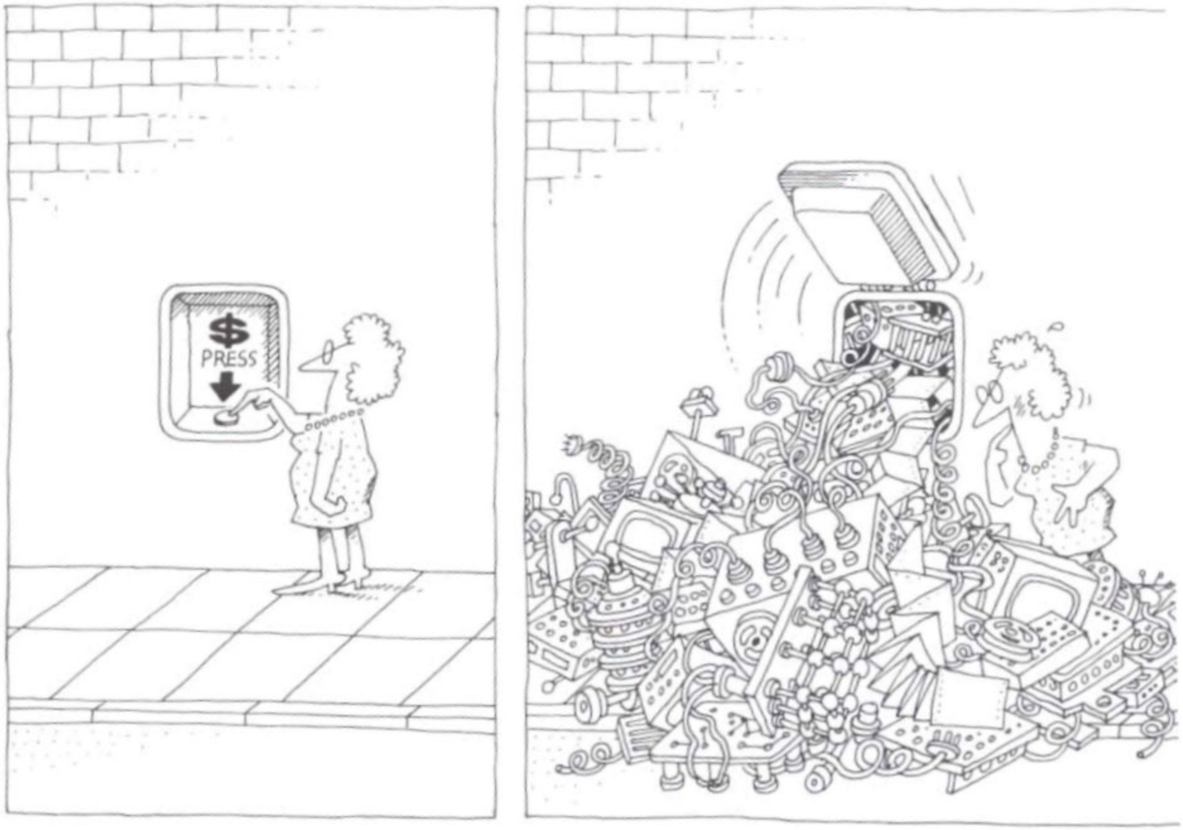
H.S. Sarjoughian

CSE 460: Software Analysis and Design

School of Computing, Informatics and Decision Systems Engineering
Fulton Schools of Engineering

Arizona State University, Tempe, AZ, USA

Creating Illusion of Simplicity



The task of the software development team is to engineer the illusion of simplicity.

ATM Machine

Source: *OOAD*

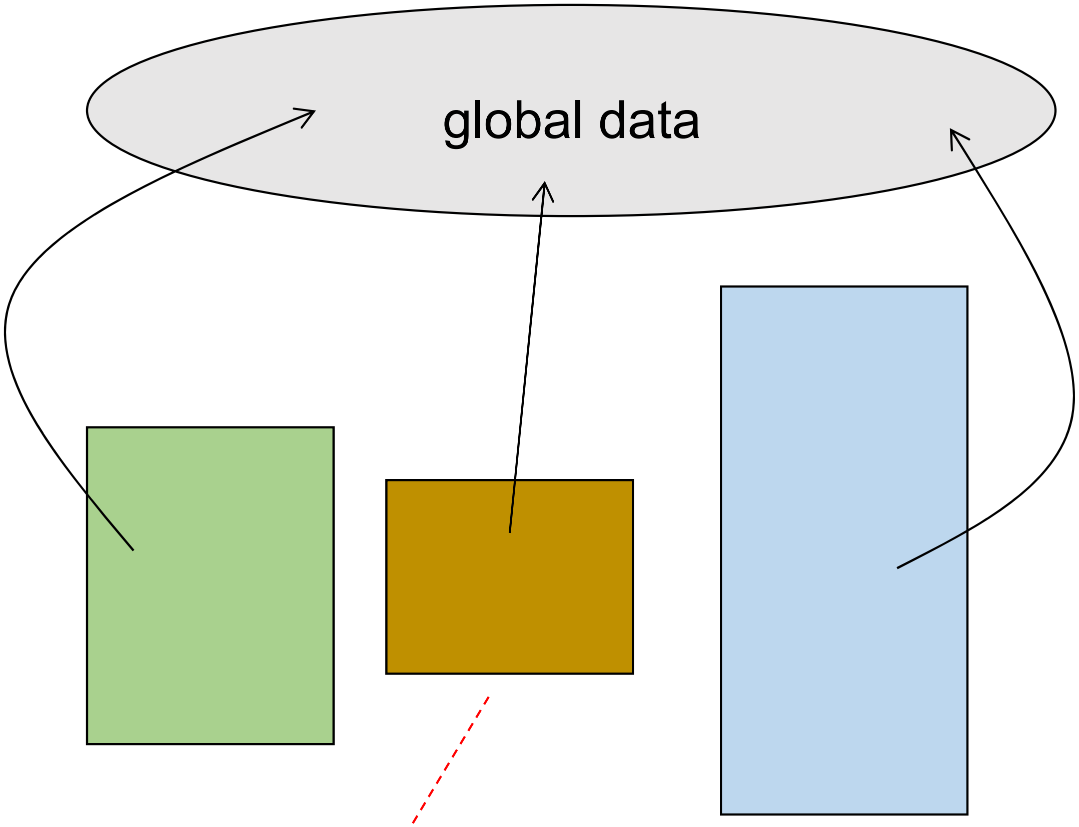
software engineers must deal with **arbitrary complexity** for building industrial-strength software

Road to the Object Model

Since early 1970s, many areas of study have collectively contributed to the development of the object-orientation/object model:

- Programming languages and methodologies (Simula 67, Smalltalk, Objective C, C++, Eiffel, Java, CLOS, …)
- Operating systems
- Computer hardware architecture
  - Better error detection
  - Improved execution efficiency
  - Fewer instruction types
  - Simpler compilation
  - Reduced storage requirements
- Databases (entity relationship diagrams)
- Artificial intelligence (theory of frames, society of agents, …)
- Philosophy and cognitive science

1st- & Early 2nd-Generation Prog. Lang. Topology



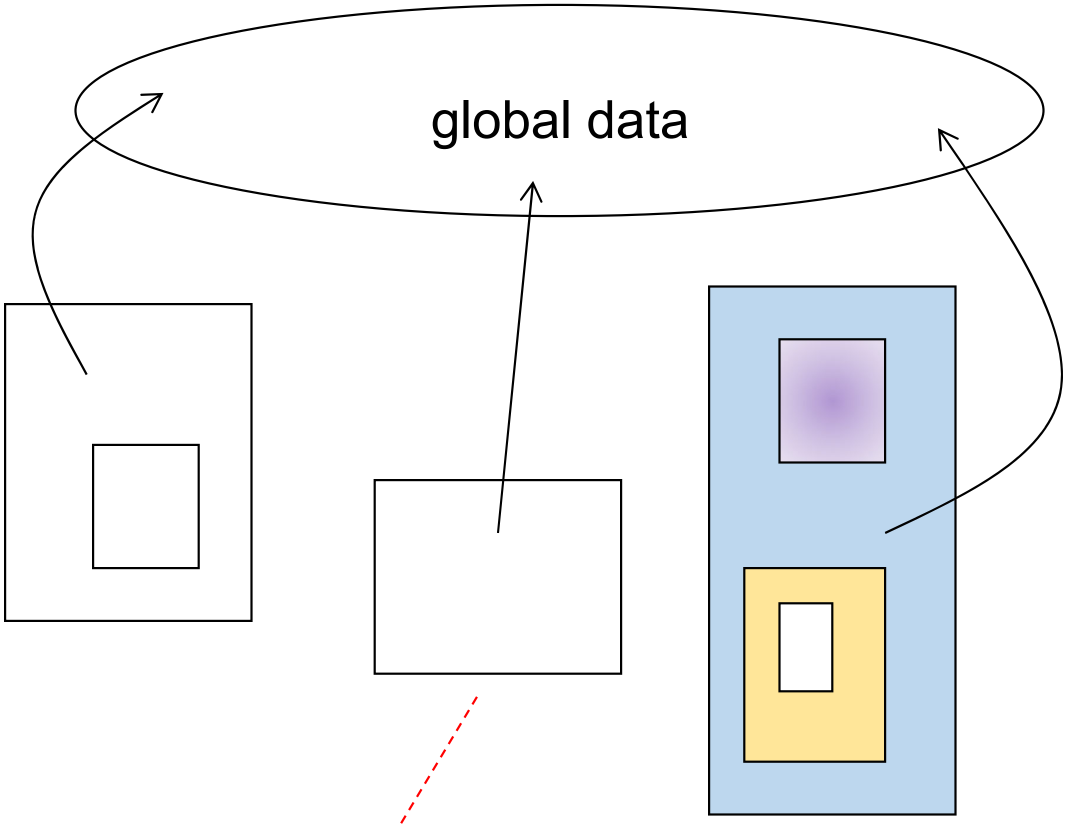
Examples:

- FORTRAN I, ALGOL 58, …

➔ data and algorithms are intertwined in a global setting

subprogram: basic building block

Late 2nd - & Early 3rd Generation Prog. Lang. Topology



subprogram: basic building block

Examples:

- FORTRAN II, LISP, …

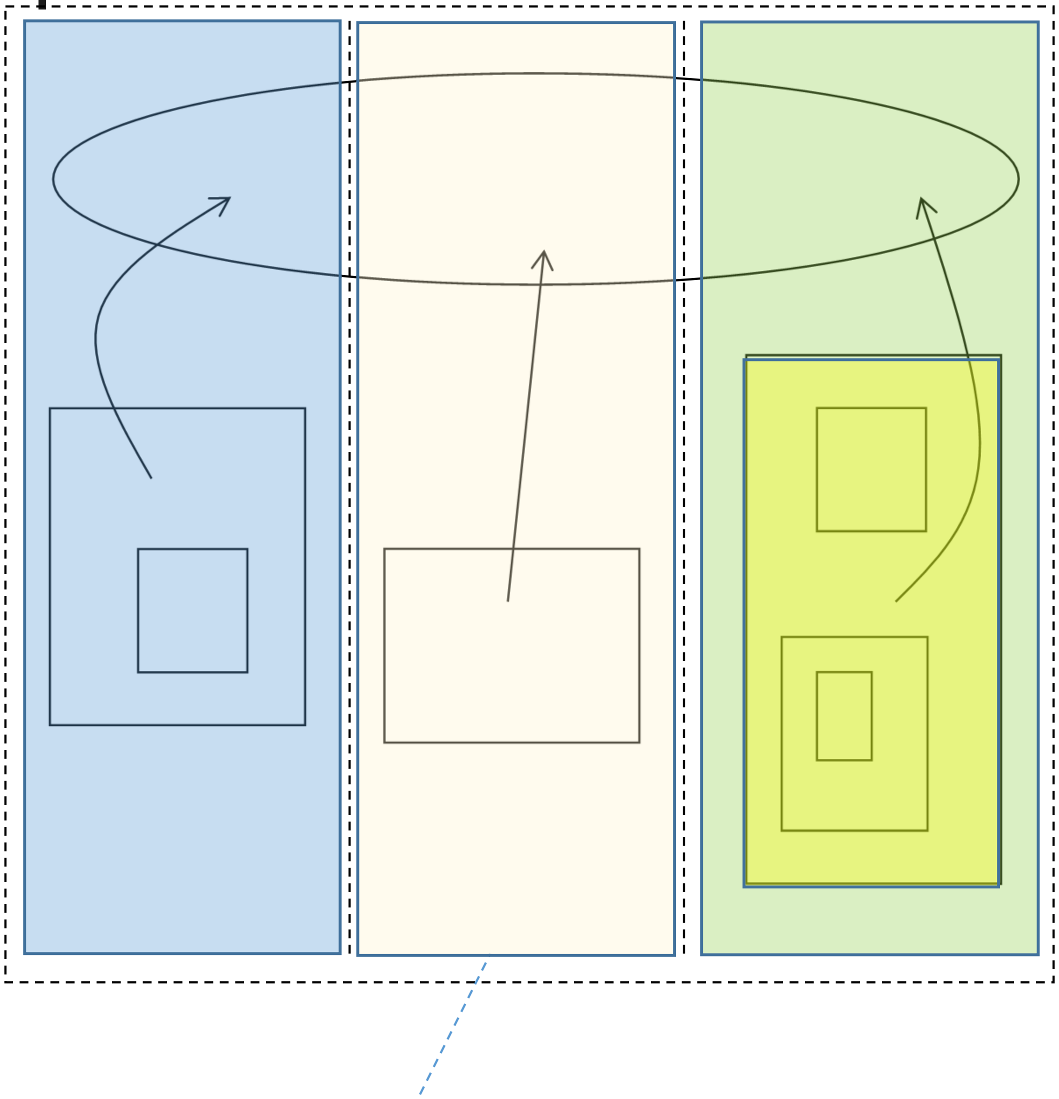
Features:

- subprogram nesting
- control structure
- abstractions

➔ structured design

➔ does not support data design and large-scale software development

Late 3rd-Generation Prog. Lang. Topology



module: basic building block

Examples:

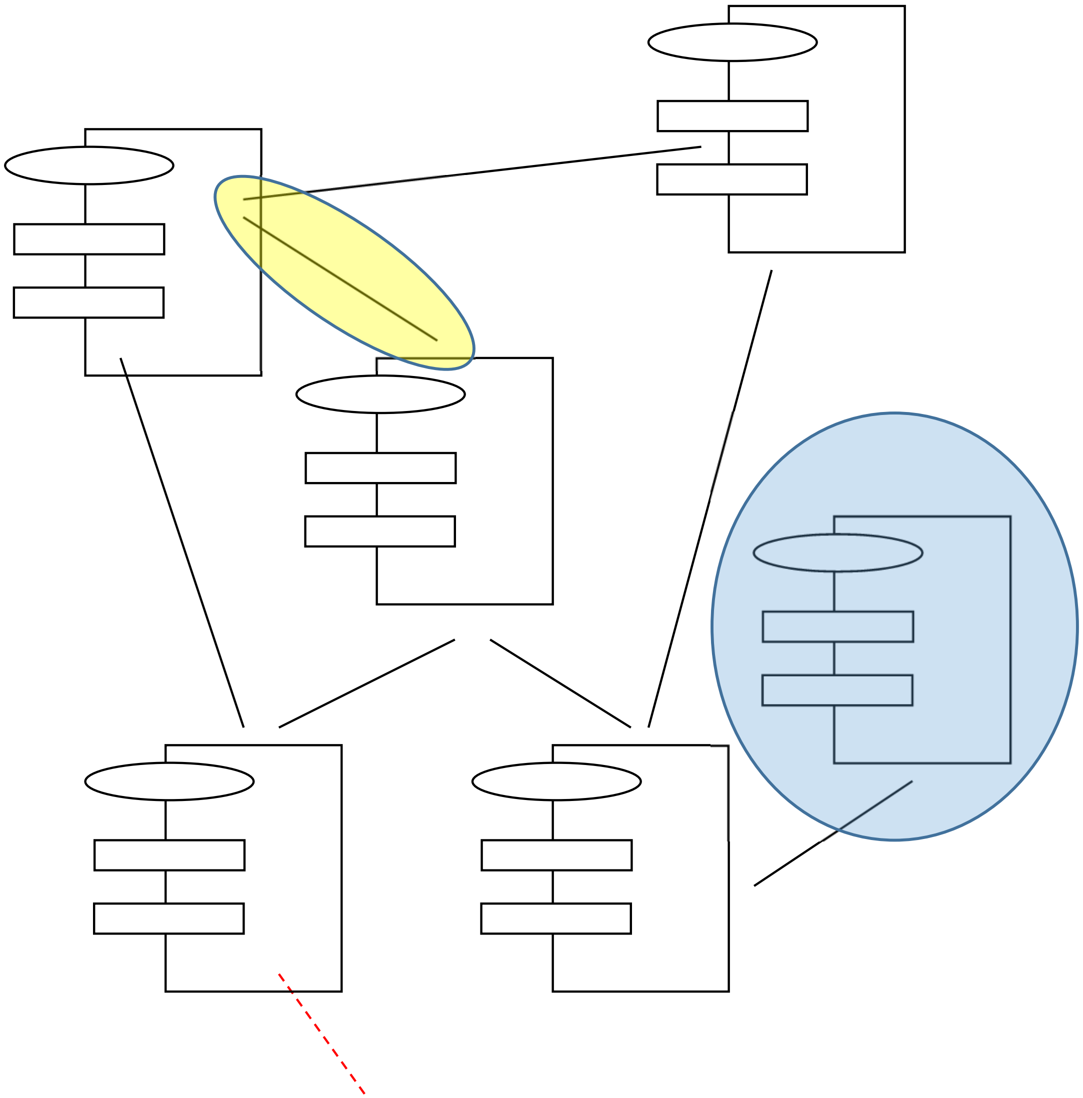
- ALGOL 68, Pascal, Simula, …

Feature:

- modularity – supports independent development of a large programming projects

➔ rules for modular structure were based on poor semantic consistency. E.g., function arguments of a subprogram could be different depending on the calling module – i.e., two modules would be built using different argument types since strong typing and data abstraction were not supported!

Object-based/Object-Oriented Prog. Lang. Topology

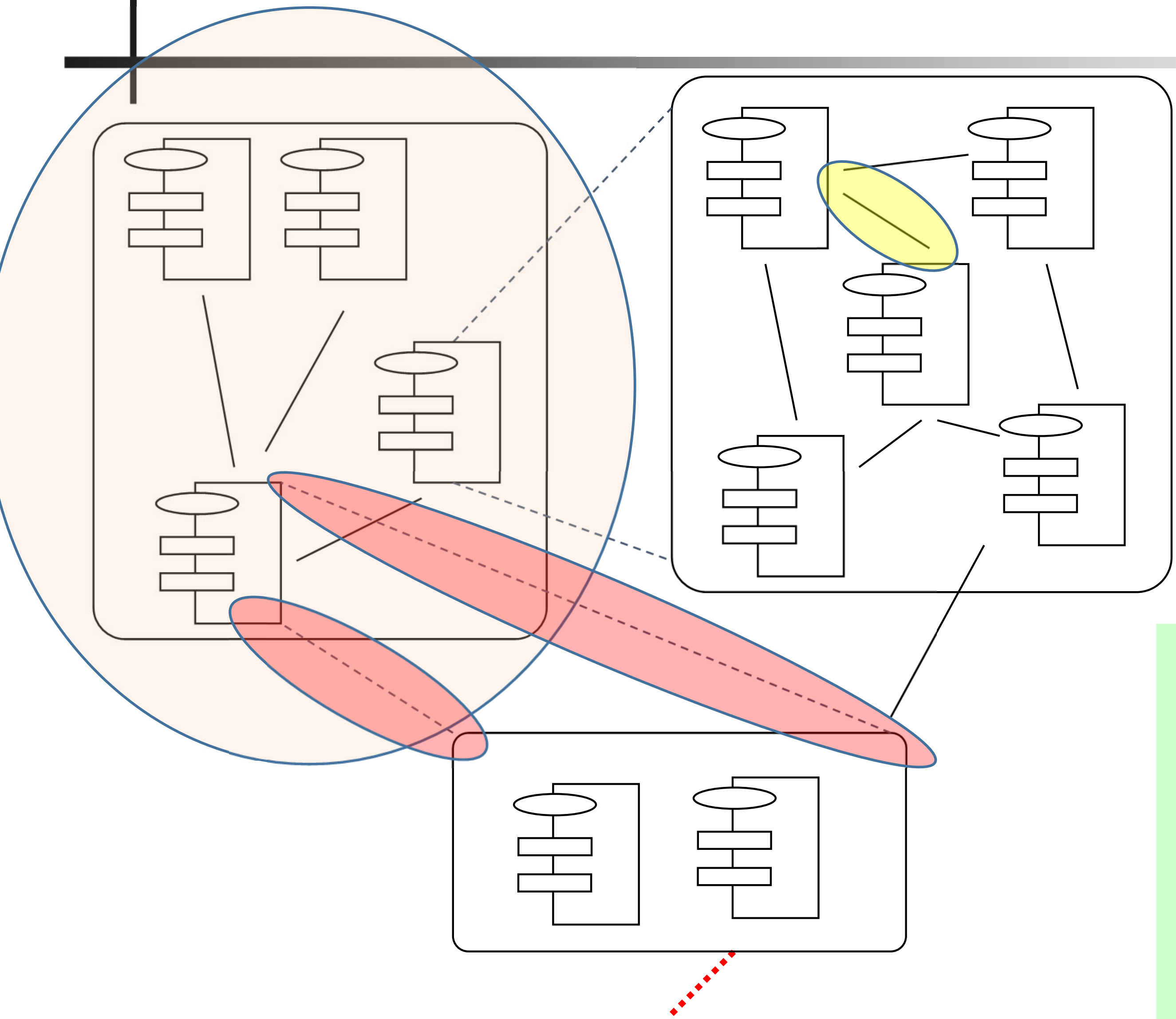


objects and classes: basic building blocks

Examples:
- Java, C++, …

Usage:
- small- to moderate-sized applications

Object-based/Object-Oriented Prog. Lang. Topology



Examples:

- C++, Eiffel, …

Usage:

- large applications

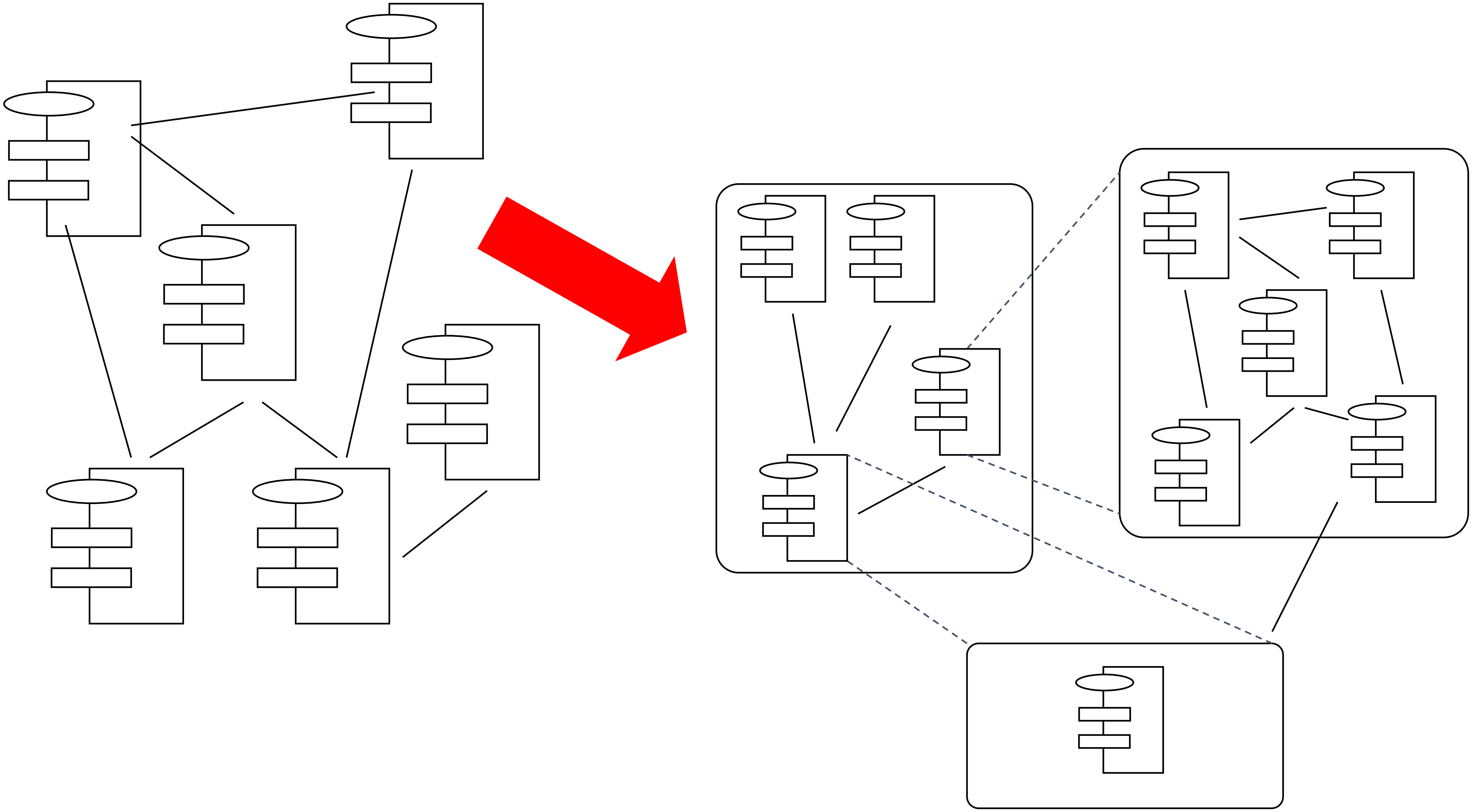
○ many systems are complex

○ modeling in the small has limited use

○ modeling in the large is increasingly a necessity

layers (collection of classes and objects): basic building blocks

Moving from Low- to High-Level Analysis & Design



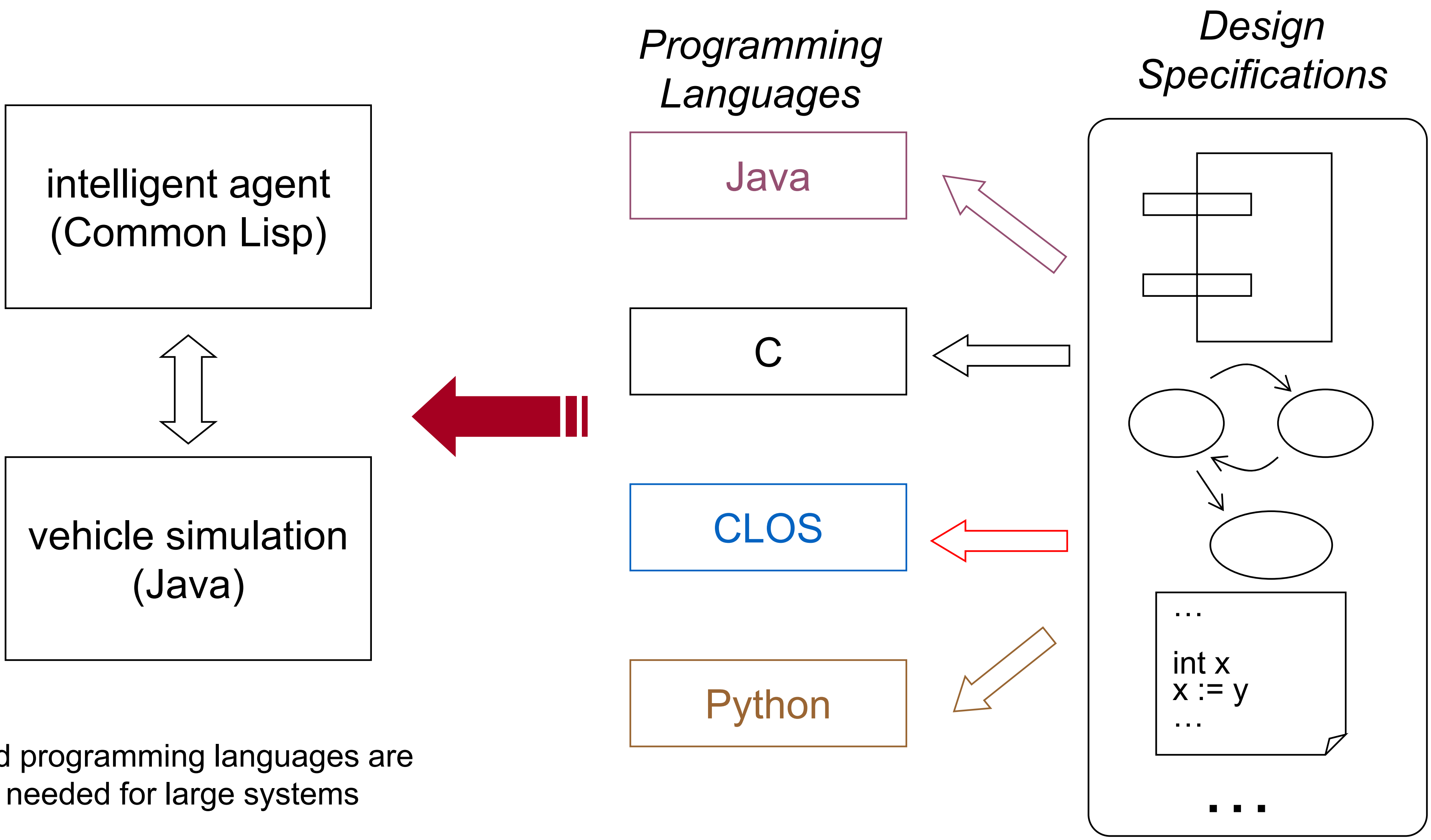
Programming Types

- Procedural-oriented – algorithms
- Object-oriented – classes and objects
- Logic-oriented – goals expressed in predicate calculus
- Rule-oriented – if-then rules
- Constraint-oriented – invariant relationships

➔ No single programming style is best for all purposes

➔ Object-orientation serves as the foundation for software architecture frameworks including Service-Oriented Computing

Specifications and Programming Languages



mixed programming languages are often needed for large systems

What Is The Object Model?

The Object model is the collection of principles that form the foundation of object-oriented analysis and design.

Object model provides a paradigm for software engineering emphasizing the principles of:

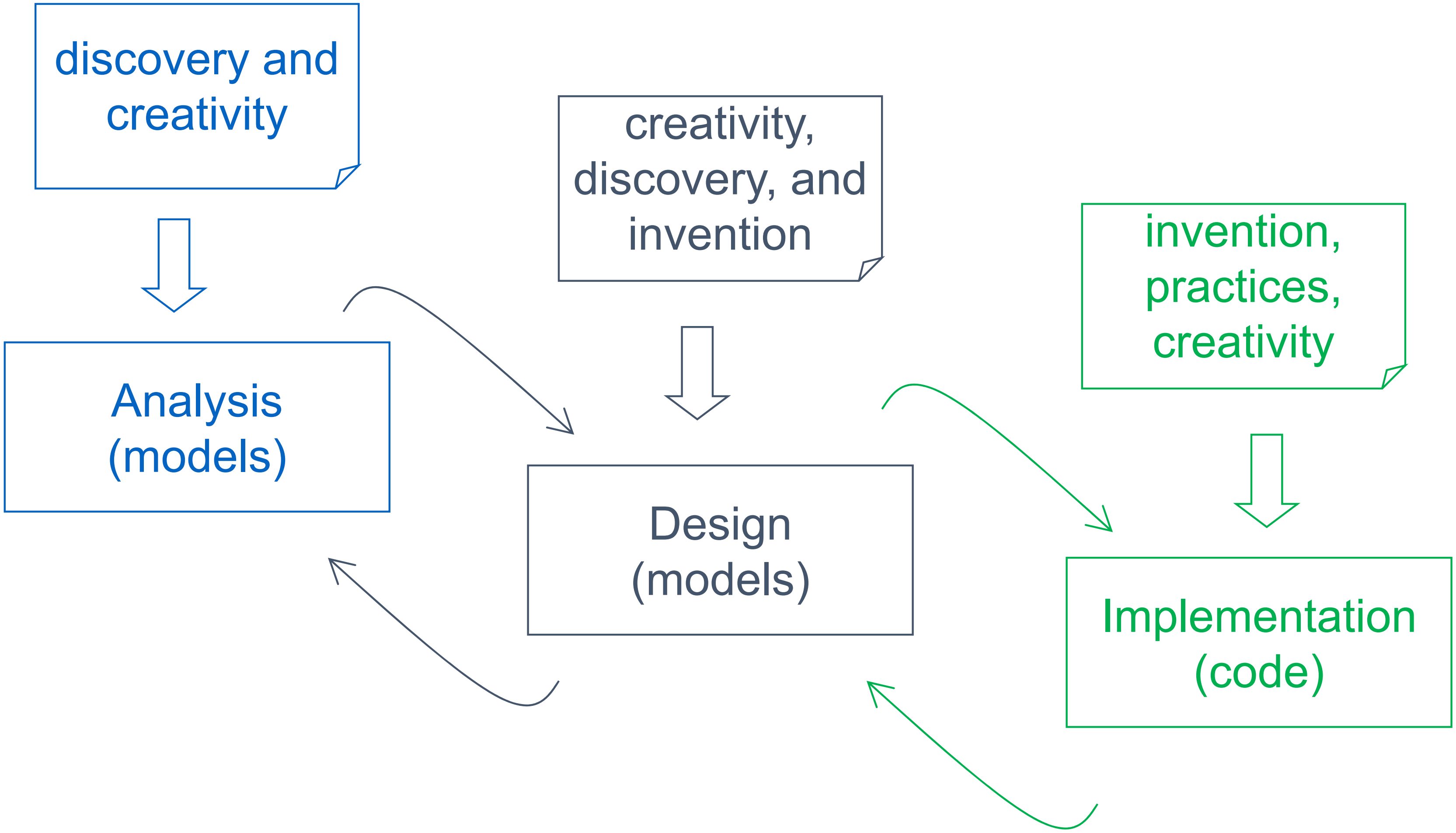
- Abstraction
- Encapsulation
- Modularity
- Hierarchy

basic principles

- Typing
- Concurrency
- Persistence

advanced principles
[will be discussed after Chapter 4, OOAD]

Discovery, Creativity, …



discovery and creativity

Analysis (models)

creativity, discovery, and invention

Design (models)

invention, practices, creativity

Implementation (code)

Object-Oriented Analysis

Object-Oriented Analysis (OOA) is a method of analysis that examines requirements from the perspective of the **classes** and **objects** found in the **vocabulary of the problem domain**.

Analysis, in part, entails:

- Requirements elicitation from end-users and stake-holders such as customer/buyer and software engineering team
- Examination of multiple views and perspectives of *what* is to be achieved from functional behavior, structural configuration, and data – primarily from the point of view of the problem as opposed to its solution!
- Creation of specifications – typically models of various sorts such as class diagrams – for use in design and other phases of the software engineering process.

further detailed treatment to follow …

Object-Oriented Design

Object-Oriented Design (OOD) is a method of design encompassing the process of object-oriented decomposition and a notation for depicting both *logical* and *physical* models from the *static* and *dynamic* aspects of the system under design – provides the vocabulary for **design solution**

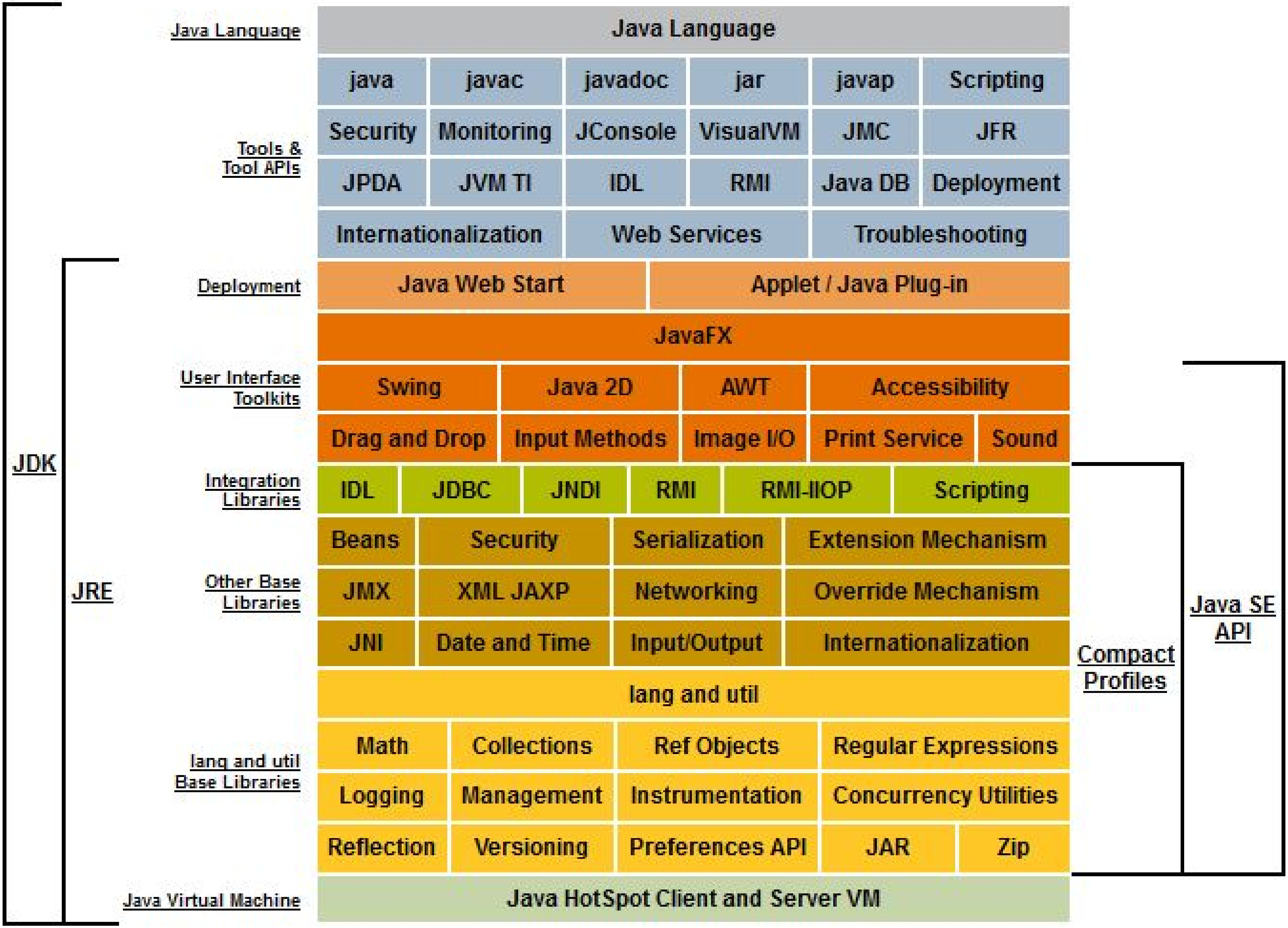
Design, in part, entails:

- Development of design – based on analysis specifications – describing *how* the intended behavior is to be achieved based on structural organizations generating compound behavior and algorithms generating primitive behavior
- Specification of data types and organization
- Creation of specifications – typically models of various sorts such as class and state transition diagrams – for implementation and other phases of the software engineering process.

further detailed treatment to follow …

References

- *Object-Oriented Analysis and Design with Applications, 2nd Edition, G. Booch, Benjamin Cummings, 1994*
- *Eclipse Modeling Framework, http://eclipse.org/*



JDK

JRE

Java Language
Java Language

Tools & Tool APIs
java | javac | javadoc | jar | javap | Scripting
Security | Monitoring | JConsole | VisualVM | JMC | JFR
JPDA | JVM TI | IDL | RMI | Java DB | Deployment
Internationalization | Web Services | Troubleshooting

Deployment
Java Web Start | Applet / Java Plug-in

User Interface Toolkits
JavaFX
Swing | Java 2D | AWT | Accessibility
Drag and Drop | Input Methods | Image I/O | Print Service | Sound

Integration Libraries
IDL | JDBC | JNDI | RMI | RMI-IIOP | Scripting

Other Base Libraries
Beans | Security | Serialization | Extension Mechanism
JMX | XML JAXP | Networking | Override Mechanism
JNI | Date and Time | Input/Output | Internationalization

lang and util Base Libraries
lang and util
Math | Collections | Ref Objects | Regular Expressions
Logging | Management | Instrumentation | Concurrency Utilities
Reflection | Versioning | Preferences API | JAR | Zip

Java Virtual Machine
Java HotSpot Client and Server VM

Compact Profiles

Java SE API