INTRODUCTION TO JAVA

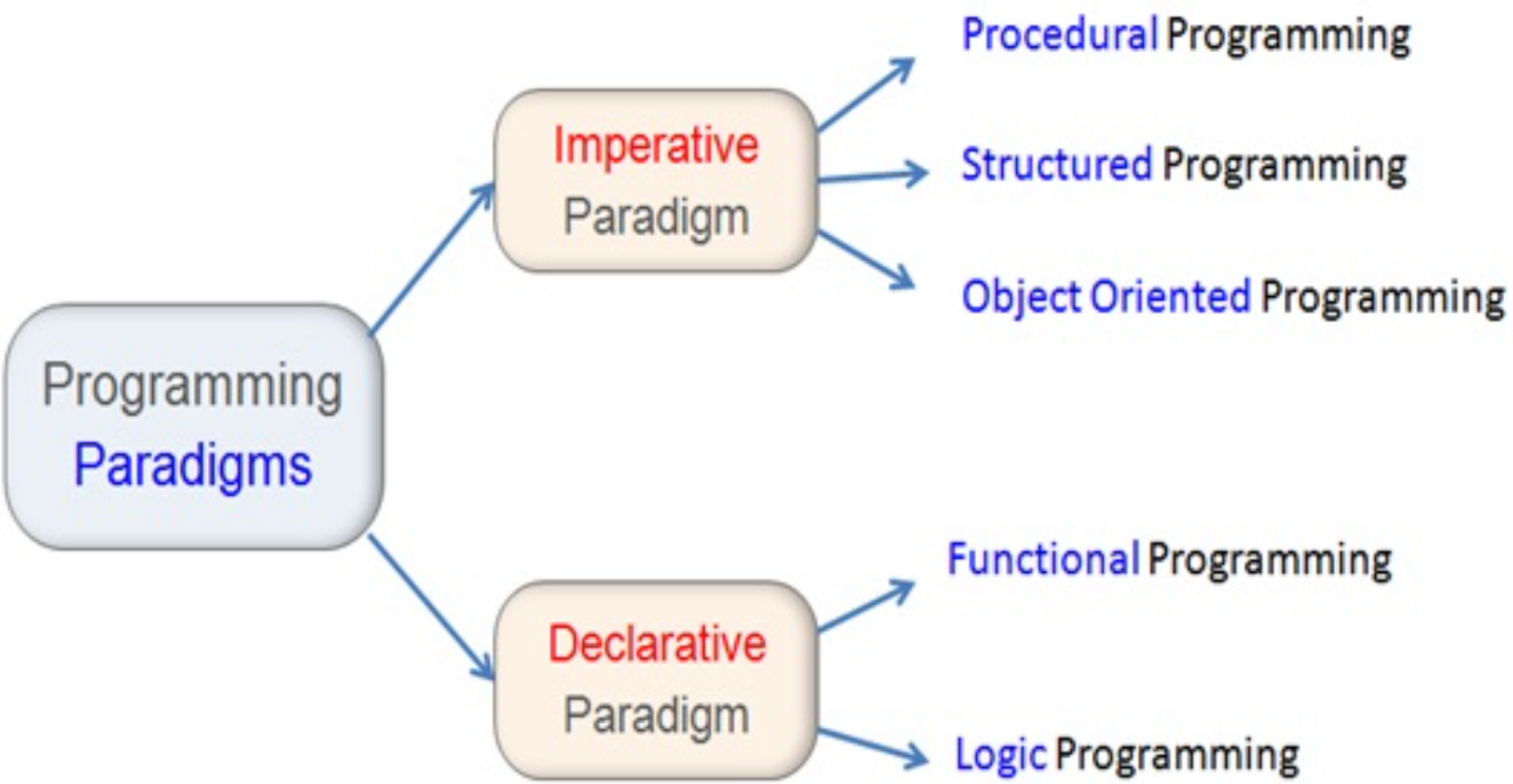
Java programming fundamentals

- Introduction to Java
- Overview of JDK/JRE/JVM
- Java Language Constructs
- Object Oriented Programming with Java
- Exception Handling

JAVA BACKGROUND AND HISTORY

Intro to Programming Language Paradigms

Programming paradigms are a way to classify programming languages based on their features
Imperative Paradigm - programmer instructs the machine how to change its state
Declarative Paradigm - programmer declares properties of the desired result, but not how to compute it



What is Java and it’s Background?

Java is a high-level object-oriented programming language with platform independent deployment.

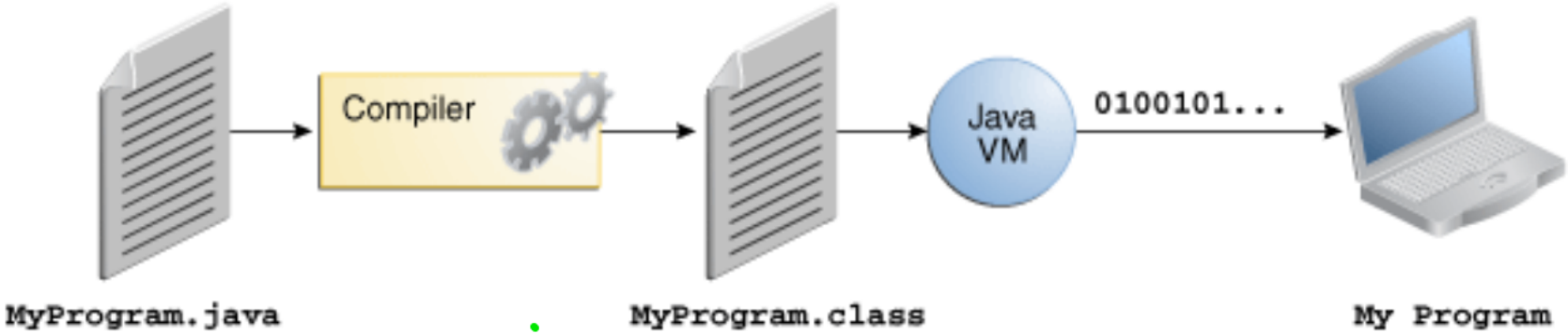
- Project started on 1991 by Sun Microsystems
- Developed by James Gosling with support from Mike Sheridan, Patrick Naughton
- v1.0 released on 1996
- JVM become open source on 2006/07 under FOSS (Free & Open Source Software)
- Oracle acquired Sun Microsystems and become owner of Java on 2009/10
- Latest version 23 and LTS versions are 8, 11, 17 and 21

Java Design Goals

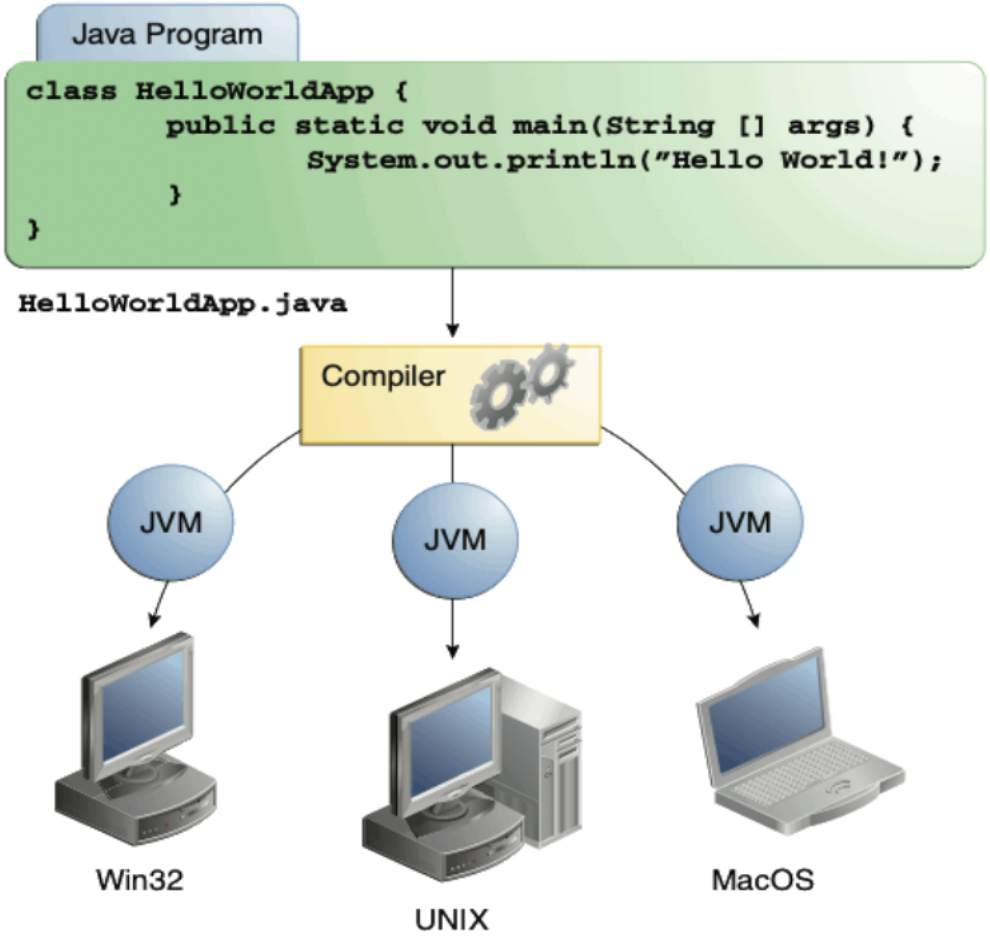
- simple, object oriented, familiar
- robust and secure
- architectural neutral and portable
- high performance (JIT)
- interpreted, threaded and dynamic

Java Characteristics / Features

- Simple
- Object oriented
- Distributed
- Multithreaded
- Dynamic
- Architecture neutral
- Portable
- High performance
- Robust
- Secure



Java is Platform Independent



Java Release History

- v1.0 -> 1996
- v1.1 -> 1997
- v1.2 -> 1998 => J2SE, J2EE, J2ME
- v1.3 -> 2000
- v1.4 -> 2002
- v5.0 -> 2004 => JSE, JEE, JME
- v6.0 -> 2006
- v7.0 -> 2011
- v8.0 -> 2014 (LTS) => OOP + FP (Lambda Expr + Stream API)
- v9.0 -> 2017
- v10 -> 2018(Mar)
- v11 -> 2018(Sep) (LTS)
- v12 -> 2019(Mar)
- v13 -> 2019(Sep)
- v14 -> 2020(Mar)
- v15 -> 2020(Sep)
- v16 -> 2021(Mar)
- v17 -> 2021(Sep) (LTS)
- v18 -> 2022(Mar)
- v19 -> 2022(Sep)
- v20 -> 2023(Mar)
- v21 -> 2023(Sep)
- v22 -> 2024(Mar)
- v23 -> 2024(Sep)

Java Flavors

- Java SE (Standard Edition)
- Java EE (Enterprise Edition) / Jakarta EE - Servlet, JSP, EJB, JAX-RS, etc..
- Java ME (Micro Edition)

Java Benefits

- **Get started quickly**
- **Write less code**
- **Write better code**
- **Develop programs more quickly**
- **Avoid platform dependencies**
- **Write once, run anywhere (WORA)**
- **Distribute software more easily**

JAVA KEYWORDS

Java Keywords

abstract	default	for	new	sealed	transient
assert	do	if	non-sealed	short	try
boolean	double	implements	package	static	var
break	else	import	permits	strictfp	void
byte	enum	instanceof	private	super	volatile
case	exports	int	protected	switch	while
catch	extends	interface	public	synchronized	
char	final	long	record	this	
class	finally	module	requires	throw	
continue	float	native	return	throws	

JAVA BASICS

Language Basic Constructs

- Data Types
- Variables
- Constants
- Operators
- Expressions, Statements, Blocks
- Control Flow Statements
- Loop Statements
- Branching Statements
- Naming Conventions
- Comments
- Arrays
- Strings

Object Oriented Programming and Related Concepts

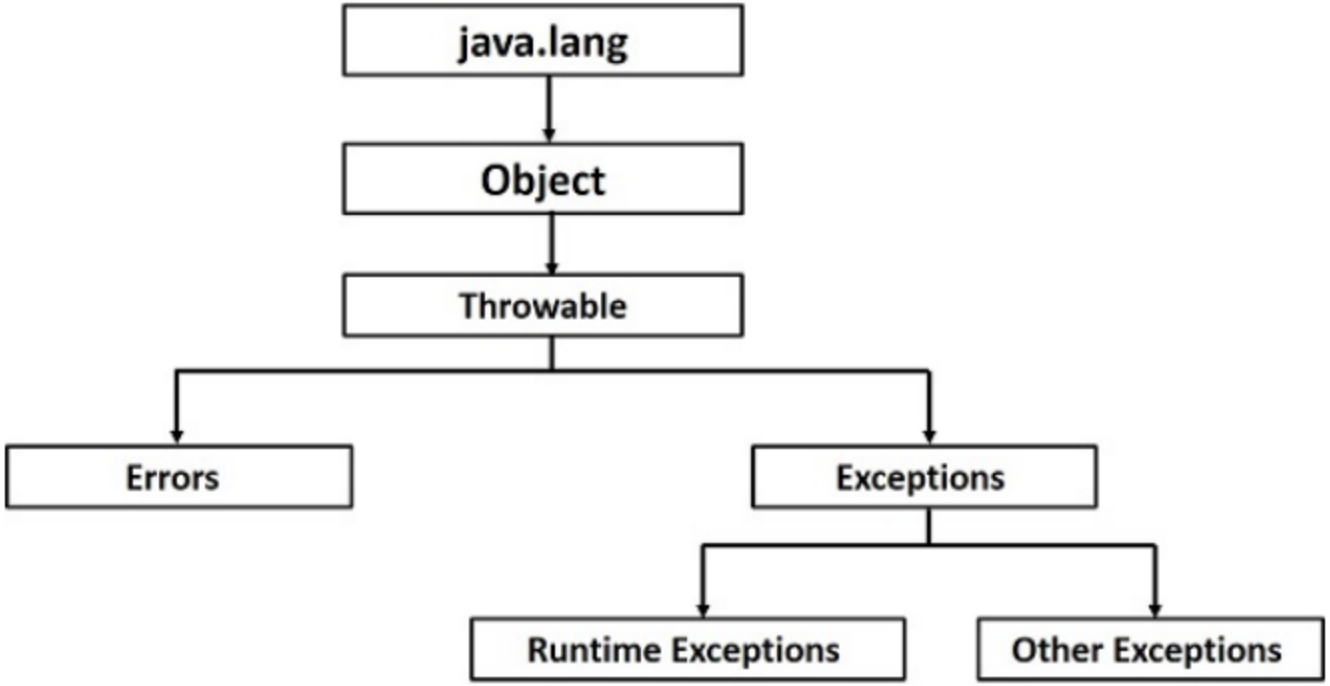
- Class
- Object
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

- Interface
- Package
- Wrapper Classes
- Object Class
- Methods
- Access Modifiers

Exception Handling

- Method call-stack and Exception
- Exception Hierarchy
- Exception vs Error
- Checked vs Unchecked Exception
- try…catch..finally block
- throws
- throw
- Custom Exception

Exception Hierarchy



JAVA COLLECTION FRAMEWORK

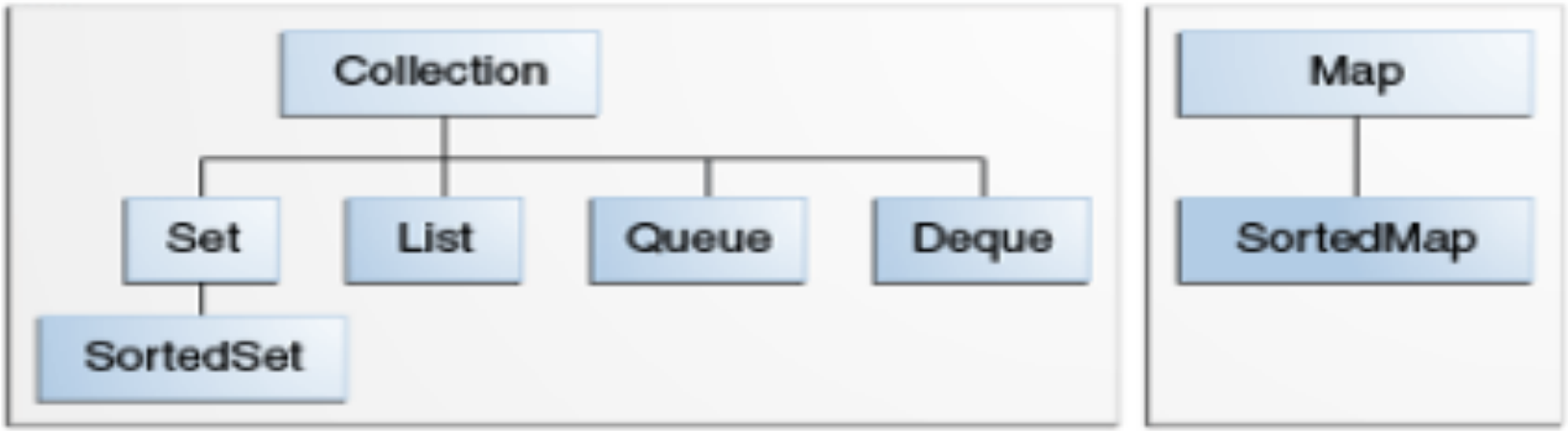
Collections Framework Overview

- A *collection* — sometimes called a container — is simply an object that groups multiple elements into a single unit.
- Collections are used to store, retrieve, manipulate, and communicate aggregate data
- A collections framework is a unified architecture for representing and manipulating collections. It consists of
  - Interfaces
  - Implementations
  - Algorithms

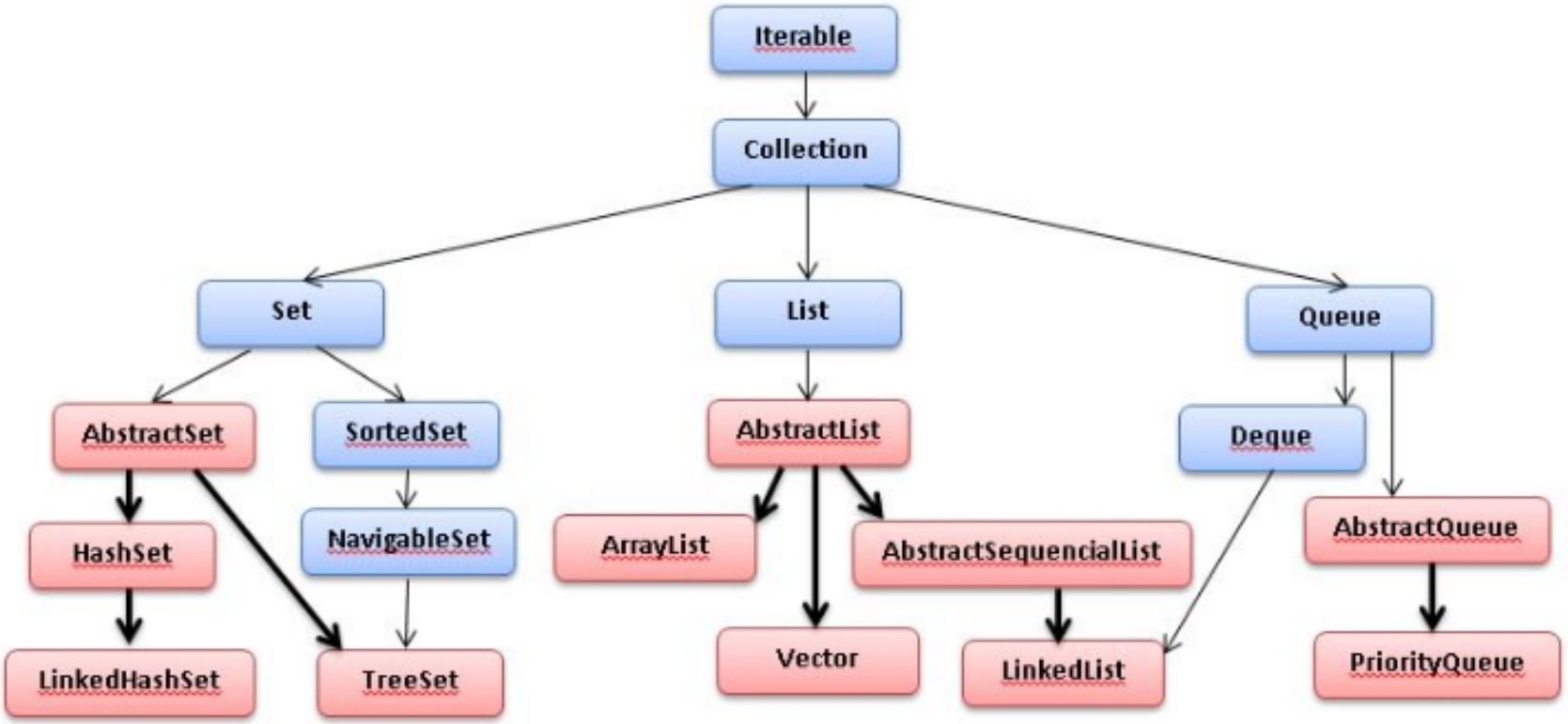
Collections Framework Benefits

- Reduces Programming Effort
- Increases Program Speed and Quality
- Allows interoperability among unrelated APIs
- Reduces effort to learn and to use new APIs
- Reduces effort to design new APIs
- Fosters software reuse

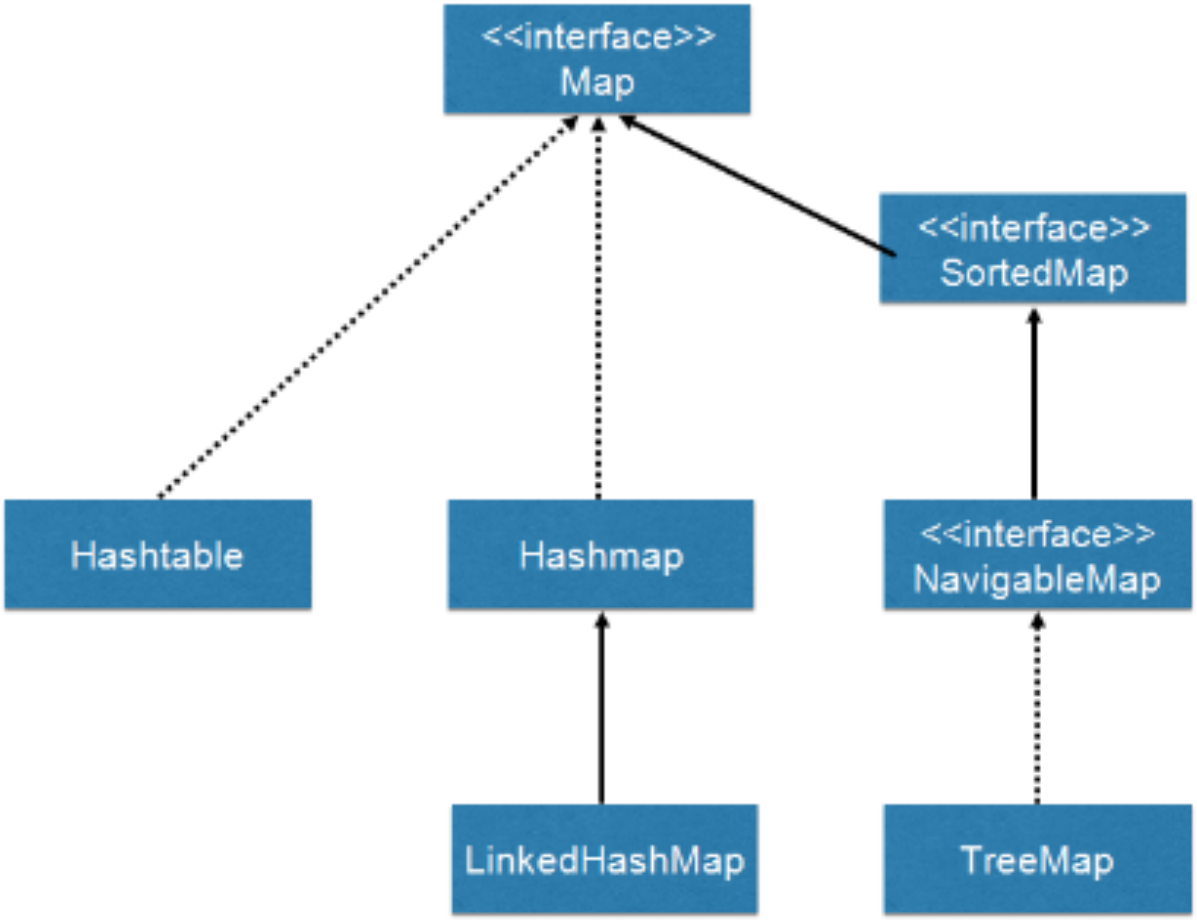
Collection Hierarchy (Interfaces)



Collection Hierarchy (Implementations)



Collection Hierarchy (contd.)

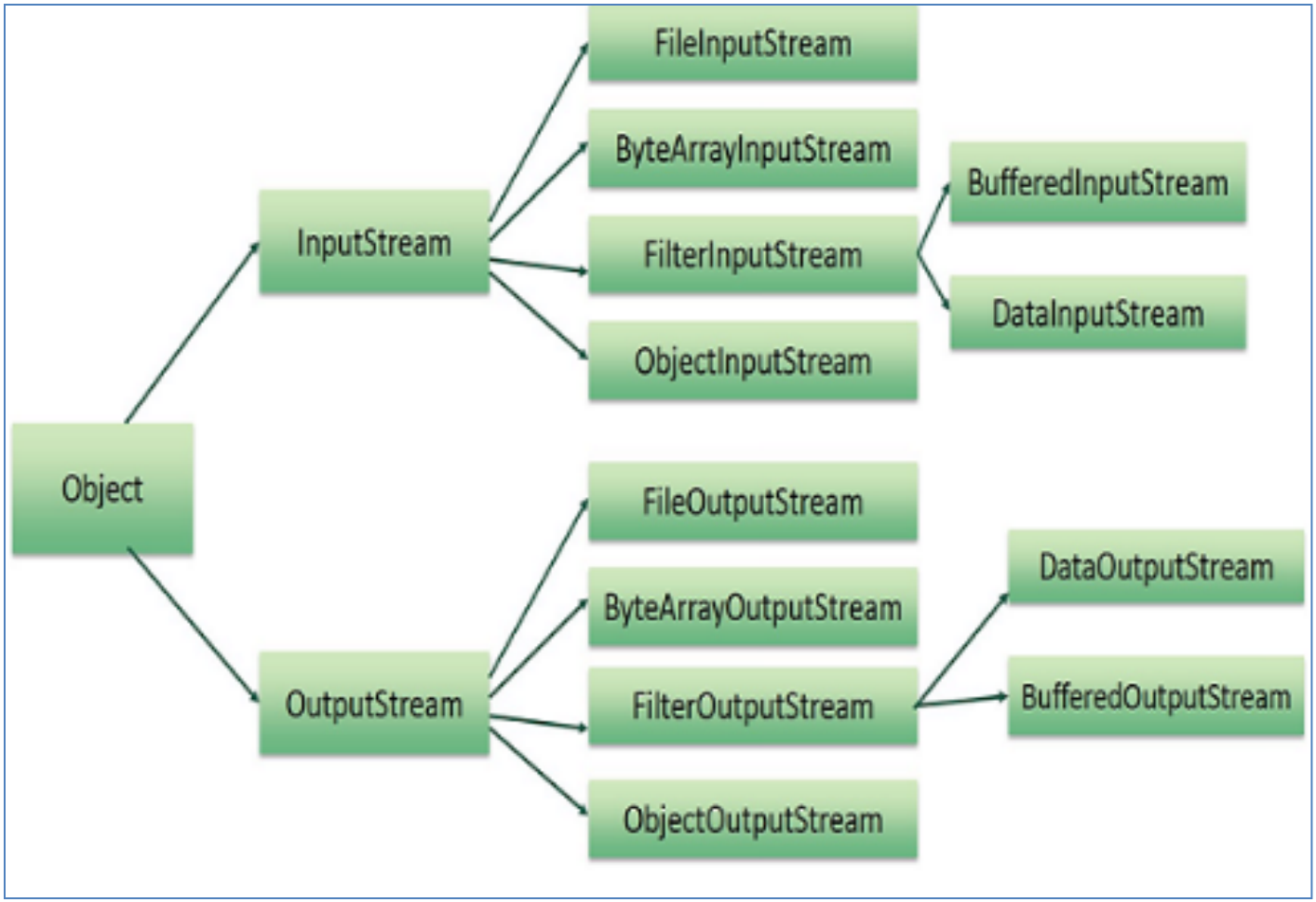


JAVA IO

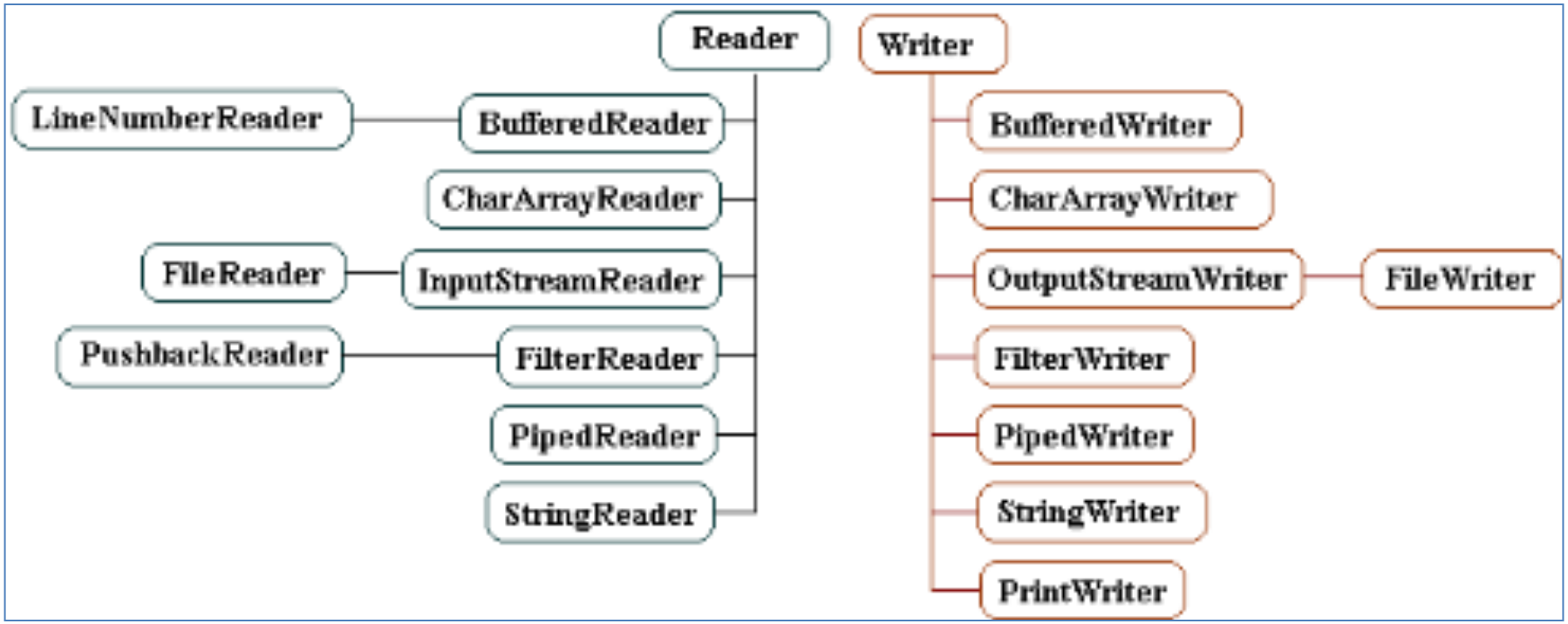
Java Serialization and I/O

- Serialization Overview
- I/O Streams Overview
- NIO (Non-blocking I/O Overview)

Byte Stream Hierarchy



Character Stream Hierarchy



Thank You!