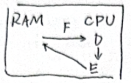
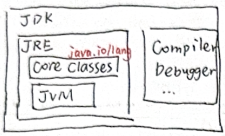
User1 User2 User3
Software Application Software ✗ OS
System Software ✗ browser, media player
Hardware CPU RAM Disk



Fetch-Decode-Execute Cycle

Java program → Java compiler → Java Bytecode
→ JVM (Class Loader : .class/.jar → Runtime Data Area
info 和 constant pool
Class-level: Method area ; Objects/instances: Heap ; Local varible
Stack → Execution Engine) references → Stack Area



Java Runtime Environment
Java Development Kit

Encapsulation Program should interact with object data only through object's method / achieved by Access Control mechanism
Inheritance Object class parent of all classes
Abstraction
Polymorphism { Compile-time overloaded methods same name / Runtime Polymorphism

Mapping name of method → final implementation
Binding { Static (compile time) method overloading 同名，参数不同
dynamic (execution time) method overriding
Constructors ✗

Software Design Principle
- High Cohesion 高内聚
- Low Coupling 低耦合 { Caller callee
- Information Hiding 信息隐藏 { data representation

Generics JDK5.0 E/T extends element non-primitive type
List<E> : Generic type ; E: Formal type parameter ; List: raw type
List<String> : Parameterized type ; String: actual type parameter
Generic Classes . List , Queue , Set
Generic Interfaces , Generic Methods
Type safety
not required type cast
Compile-time checking
Use "?" to create a relationship between generic type
Unbounded : Pair<?> Upper Pair<? extends T> Lower Pair<? super T>

Collection ← Iterable List, Queue, Set
An Iterable class could be iterated over using an Iterator
Public static void removeNulls (Collection<?> c){
for(Iterator<?> i = c.iterator(); i.hasNext();){
if (i.next() == null){ i.remove();}}

Interfaces Implementations Algorithms
Array
List: Array LinkedList: doubly linked list
hashMap: converts internal address of object into an integer
Override equals → override hashCode
LinkedHashMap : insertion order ; TreeMap : ordered . HashSet: HashMap

Functional programming is a programming paradigm 范式 函数式编程
OOP : organize data and logics in objects (fileds and methods)
procedural programming : organize code as sequential procedures
functional programming: organize code as functions. behaviors separated
Functions can be passed as arguments. assign to variable. be returned
writeFile (filename) ——— Side Effects ——— External file change
pure function produce same output for the same input; x assign
Immutability Varibles once defined, never change their value
Lambda Expression Java8 Anonymous function with no name/identifier
(param1, param2) → { expressions or statements }
function parameters function body multi(;) return can✓
Collections.sort (strList, (s1,s2) → Integer.compare(s1.length(), s2.length()))
Collections.sort (strList, new Comparator<String>(){
public int compare (String s1, String s2){
return Integer.compare(s1.length(), s2.length());}});
List (raw type) → List.length() ✗ → List<String> parameterized type
1. Create & instantiate a functional interface
public interface a{ double getPivalue();}
a ref = () → 3.145;
ref.getPivalue.

Ⅱ Executing same operation when iterating elements
List<String> str = new ArrayList<String>();
Str.forEach (elem → sout(elem));
Method reference : refer to this one method by name
MyInter ref = s → sout(s); lambda expression
MyInter ref = System.out::println; method reference
Static/Instance method (Bound/Unbound)/constructor
1 Static ClassName :: staticMethod str → Integer.parseInt(str)
2 InstanceName::instance Method System.out::println
3 ClassName:: instanceMethod (s) → s.toLowerCase() String::toLowerCase
4 ClassName::new PersonIJ::new Size → new PersonIsize]

java.util.Stream Java 8 process collections of objects
source intermediate operations terminal operation
Lazy evaluation, all IO do not get exearted until terminal oper
AnyMatch() allMatch() noneMatch() collect() count()
findAny() findFirst() forEach() min() max() reduce()
toArray() returns a non-stream type
Terminal operations are eagerly executed
Optional Java SE8 prevent INull Pointer Exception
optional. empty
not null

I/O and File in java.io packer
Input & Output Byte Stream & Character Stream i18n
The International Morse Code 1837 A-Z.num.char
Chinese Telegraph Code 1872 Ch char → 4 digit num 0000 9999
ASCII 7bits → 128 characters extend 8 bits → 256
GB2312, 简 2bytes → GBK → GB18030
Unicode code point 1114112 0hex → 10FFFFhex
UTF-8 minimum 1 byte. UTF-16 min 2bytes x 3byte √ 4bytes
UTF-32 always 4 bytes not compatible with ASCII table

Unicode Transformation Format - 8 bit . 1-4 bytes

1st Byte	2nd B	3rd B	4th B	num free bits
0xxxxxxx				7
110xxxxx	10xxxxxx			5+6=11
1110xxxx	10xxxxxx	10xxxxxx		4+6+6=16
11110xxx	10xxxxxx	10xxxxxx	10xxxxxx	3+6+6+6=21

eg Unicode U+6C49 → 01101100 01001001 16 bit
11100110 10110001 10001001
Java Char 16bit unsigned int U+0000 - U+FFFF, code points
Supplementary characters >U+FFFF. a pair of char values 4 bytes
Stream Continuous flow of data sequentially accessed
Container linked to a data source, and a data destination
Byte Streams Input/Output inconvenient for processing info
Stored in Unicode
abstract classes
Character Stream Separate hierarchy provides classes, inheriting from Reader and Writer, read & write base on char value
Subclasses impl read() / write xxx Stream / xxx Reader
FileReader specify encoding scheme UTF-8 √
FileRead GBK
FilterInputStream +gzip func +buffered
InputStream zfile = new GZIPInputStream(bfile)
InputStream bfile = new BufferedInputStream(file)
InputStream file = new FileInputStream("src/")
+pushback func int b = pbin.read pbin.unread(b)
pushbackInputStream pbin = new PBIS(new BIS(newFi)
+read-numbers func +zip readDouble() readInt()
DIS din = new DIS(new ZIS(new FIS("")))
File inputFile = new File("input.txt")
Scanner in = new Scanner(inputFile)
while (in.hasNextDouble()) / next / nextInt
PrintWriter out = new PrintWriter("output.txt")
out.println("")
Standard Streams System.err/in/out java.lang.Object
public final class System extends Object
System.in byte stream with no character stream features
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
br.readLine.
Standard Input as a character stream

System.out → PrintStream object [redirect]
PrintStream out = new PrintStream(new File("src/..."))
System.setOut(out)

Serialization 序列化 Convert state of obj into byte stream
Data persistance · Store on disk · deserialization
Classes implement serializable interface marker/tagging [obj]
serialized → types & data fields; OOS · all fields [serial num]

Path java.nio.file.Path / Paths Path p1 = Paths.get(...)
highest elem → index 0 lowest → index [n-1]
A path starts with root is ~~relativ~~ absolute, otherwise relative
toAbsolutePath() file doesn't need exist (exception / exist)
toRealPath relative → absolute redundant → remove
java.nio.file.Files Files.list first-layer
Files.walk (depth-first) visitFile / PreVisitDirectory
FileVisitResult. CONTINUE / SKIP_SUBTREE / SIBLINGS
TERMINATE

Throwable → Error, Exception
Only Throwable or its subclasses can be thrown by JVM or the throw keyword, caught by catch
LinkageError, AssertionError, ThreadDeath, VMError internal
Mostly thrown by JVM considered fatal
Exception (checked) → RunTimeException (Unchecked)
checked exceptions cannot be ignored when compilation
finally { } always executes

Concurrent Programming: multiple computations happening same time
Process, Thread a program starts Process → multiple thread
Extend Thread (decouple tasks) / Impl (prefer) Runnable
Thread class / Runnable Interface
run() method override (shares Heap)
Runnable interface should be implemented by any class whose instances are intended to be executed by a thread.
Synchronization Concurrency API Java5
synchronized keyword JDK 1.0
ReentrantLock → Lock balanceChangeLock = new R...L() (Lock)
Avoiding DeadLock java.util.concurrent.locks.
SufficientBalanceCondition = balanceChangeLock.newCondition().
Condition every object in Java has an intrinsic lock.
Thread states. NEW RUNNABLE BLOCKED WAITING TIMED_WAITING TERMINATED (start, body .run, lock 等, join.wait)
volatile lock-free visibility add store / ~~ee~~ load
Atomicity happens all at once X provide
java.util.concurrent.atomic AtomicInteger increamentAndGet() (de)
CAS (compare and swap) read the variable (O) from (M)
New value N store check ~~New~~ O with M
Only vector and Hashtable thread-safe
CopyOnWriteArrayList Java 5 util.concurrent
Sequential writes and concurrent reads.
Collections using Lock blocking queu full add / remove empty
automatically balance workload | CAS ConcurrentLinkedQueue / SkipListMap
Callable returns value. Runnable X, both asynchronous
java.util.concurrent. ExecutorService asynchronous
ThreadPoolExecutor implement ExecutorService Callable Runnable
Future holds result of asynchronous computation. after obj

Static web pages. server-side rendered HTML fixed
Dynamic JS rendered HTML API Java.net
REpresentational State Transfer software architectural style
REQUST = Verb + Object

Graphical User Interface — Container Panel Window JComponent
Object ← Component ← AWT Component import javafx.application.Application
public class MyFX extends Application {
@override start() public void start(Stage primaryStage)
Stage Scenes Scene Graphs | minWidth resizable UNIFIED DECORATED
alwaysOnTop fullscreenExitHint maxHeight stage.initStyle
Scene → attach a stage → display one Scene

java.lang.reflect examine / modify behavior of methods, classes interfaces at run time public final Class Class<T> extends Obj
class c = x.getClass()

Collection is a interface, Collections is class
JDK > JRE > JVM
ArrayList ~~internally~~ implements the iterable interface
Abstract class is the abstraction of class. interface → behavior
A class can inherit at most one abstract class, multiple interfaces

Reflection: getting info of a class through its Class instance (dynamic class loading & introspection)
Only the JVM creates Class (no public constructor)
一个class仅有一个class对象但 Student1, 2, 3 有自己对象空间
String.class == Class.forName("java.lang.String") == "hello".getClass()
Object [Class<T> / AccessibleObject { constructor / field / Method
Annotation @ metadata semantics unchange way change treat
Compiler instructions, Build-time instr. Runtime ins
Predefined annotations: Build in, Meta-ann to other
Custom ann (element with default value 引用) Deprecated Override SafeVarargs 抑制warn
@Target type annotation @Retention how stored
@Inherited type can inherited from superclass
@Documented javadoc @Repeatable

[JavaME JavaSE JavaEE] reduce complexity multitiered application model
Client Tier client → web / application / Web Tier
Business Tier Data Tier EIS enterprise info systems
Java EE Severs host component types Servlet, EJB..
Containers provide standardized runtime environment such as concurrency management, Lifecycle m.. and request man..
Servlet respond requests, only understands objects
Browser (req) → web server (Apache HTTPD) → web container (HTTP Requests) → Request obj → Servlet (web.xml)
GenericServlet (Abstract) ...> Servlet implement · servlet container
↑ extends multithreading
HTTPServlet ← MyServlet
TomCat WebContainer running servlet & JSP
File system stores unstructed, unrelated. → database
JDBC Java DataBase Connectivity javax.sql / java.sql
ORM object-Relational Mapping XSQL → object
规范 JPA Java Persistence API standard specification for ORM
框架 Hibernate framework Persistence logic Objects. no SQL

Spring Plain Old Java Object instead of EJB core container requi...
based on IoC and Dependency Injection
Inversion of Control control of obj → container or framework
DI make a class inpendent of its dependencies
@Component automatic bean detection 实例化+关系、inject
@Autowired ~~get~~ setter constructors @IO dependency
@Configuration configuration by itself @ComponentScan
@Bean work with to create spring Beans.

AOP Aspect-Oriented Programming 面向切面编程 separation of cross-cutting concerns
business code and non-business code decoupled
Logging Security Transaction Handling 身份验证、事件回滚、和记...
SpringMVC integrated ver. Model View Control
Model define data structer View Defines displayUI Controller contains control logic
SpringBoot bootstrapping a Spring app contains everything
POM Project Object Model pom.xml. core config
Client ←HTTPRequest→ Controller ↔ Service layer (DI) ↔ model, DB, JPA / Spring Dat
Repository class Extending CRUD service
define JPA entity @Entity @Table @Id @GeneratedValue
Thymeleaf server-side Java template engine web/standalone
MVC RESTful different: how HTTP response body is created
view technology return data | as object written directly JSON
@RestController @RequestMapping("/api/...") hand HTTP request
@GetMapping @RequestParam(value = "email")
@PutMapping @PathVariable(studentId)

Logs plain text. XML HTML OFF FATAL ERROR WARN INFO DEBUG TRACE. ALL Appender Console File SMTP (org.apache.logging.log4j)
Formatter Simple. XML. HTML. log4j.prop. key-value logger LogMan
Application → SLF4J → java.util.logging / Log4j / LogBack
SB LogBack PatternLayoutEncoder INFO × TRACE DEBUG
Unit Test individual method Integration group of classes
System Test complete system Acceptance integrated system
Regression change Junit org.junit.jupiter
Test Class × abstract ✓ single constructor Test Method @Test Repeat
Lifecycle Method @BeforeAll × private
@BeforeEach before @Test ✓ void return type
× private, static | BeforeAll × private ✓ static

boolean equals (Object obj)
same physical object in memory → True
String toString()
default name of class + @ + hashCode
int compareTo(T o)
⊖ less than 0 equa ⊕ greater than
public class ArrayList <E>
public boolean add(E e)
append element to the end
public E get(int index)
returns position
public interface Iterable<T>
public interface Collection<E> extends Iterable<E>
public interface Set<E> extends Collection<E>{}
Set<Type> S1,S2; boolean isSubset = S1.containsAll(S2)
Set<Type> union = new HashSet<>(S1) union = union.addAll(S2)
intersection → retainAll difference → removeAll
List<E> extends Collection<E> E get(int index)
E set(int index, E element);
boolean addAll(int index, Collection<? extends E> c);
List<E> subList(int from, int to)
public interface Map<K,V> size, isEmpty, containsKey
containsValue(Object value) get put remove clear()
void putAll(Map<? Extends K, ? Extends V> t);
public class Collections extends Object
static <T extends Comparable<? super T>> void sort(List<T> list)
public interface Comparator<T>
compareTo(T o) default sorting order Comparable
Compare(T o1, T o2) different properties Comparator
java.util.function Consumer<T>
strList.forEach(elem → System.out.println(elem));
public ~~static~~ void forEach(Consumer<? super E> action)
Supplier<T> Supplier<String> textSupplier = () → "Hello"
Sout(textSupplier.get())
Predicate<T> list.removeIf(e → e.length() < 3)
Function<T,R> Function<Integer,int[]> a = int[]::new
int[] intArray = a.apply(5);
public interface Stream<T>
defaut Stream<E> stream() Stream<String> stream = list.stream()
static <T> Stream<T> generate(Supplier<T> s)∞
Stream<String> echos = Stream.generate() → "Echo");
Stream<Integer> natual = Stream.generate(new NatualSupplier());
natual.limit(20).forEach(System.out::println);
class NatualSupplier implements Supplier<Integer>{
int n=0;
public Integer get(){n++; return n;}}
Static <T> Stream<T> iterate(T seed, UnaryOperator<T> f)
Stream<Integer> evenNumber= Stream.Iterate(2, n→n+2)
Static <T> Stream<T> of(T... values)
Integer[] array = new Integer[]{1,2,3};
Stream<Integer> istream = Stream.of(array);
IntStream / LongStream / DoubleStream
IntStream Stream = IntStream.range(5,10)
Stream<T> filter(Predicate<? super T> predicate)
List<Integer> list = Arrays.asList(10,20,33,43,54,68);
list.stream()
.filter(element → (element%2 == 0)
.forEach(element → sout)

<R> Stream<R> map(Function<? super T, ? extends R> mapper)
List<String> strList = new ArrayList<String>();
strList.add("123");
strList.stream() function
.map(Integer::parseInt) Stream<Integer>
.forEach(System.out::println);
.sorted() / .sorted(Comparator<? super T> comparator)
pointList.stream()
.sorted((p1,p2) → p1.X.compareTo(p2.x))
.forEach(System.out::println);
Optional<T> reduce(BinaryOperator<T> accumulator)
reduce((a,b) → a+b) .get (NoSuchElem)
partial result next elem accumulator function
arrayList.stream().reduce((a,b) → a+b).orElse(0)
.reduce(0, (~~sum~~) (a,b) → a+b)
Collector<T,A,R> type result type mutable
List<String> result = stream.collect(Collectors.toList())
TreeSet<String> result = stream.collect(Collectors.toCollection
(TreesSet::new)) ‖ joined = Stream.collect(Collectors.joining(","))
Map<String, Integer> map = Stream.collect
(Collectors.toMap(Function.identity(), String::length)));
Collectors.groupingBy(Function<T, K> classifier)
Map<Integer, List<String>> group = Stream.collect
(Collectors.groupingBy(String::length));
Set<String> , Collectors.toSet()
Stream<String> stream = Stream.of("la","lbb","c","2a","2a","2bb")
Map<Character, Set<String>> group = stream.collect
(Collectors.groupingBy(S → S.charAt(0), Collectors.mapping
(S → S.substring(1), Collectors.toSet())));
Optional<String> result = Stream.findFirst();
Optional<T> class
class Computer{
private Optional<Soundcard> Soundcard
public Optional<Soundcard> getSoundcard(){} return
public void setSoundCard(Optional<SoundCard> soundcard)
this.soundcard = soundcard;}
class Soundcard{ private Optional<USB> usb;
class USB{ String version}
public static String getUSBversion(Computer computer){
return Computer.getSoundcard()
.flatMap(SoundCard::getUSB)
.map(USB::getVersion)
.orElse("UNKNOWN")
try(InputStream input = new FileInputStream("src/"))
int n; while((n=input.read())!=-1){}}
Work "Hello World" ✕ "计算机" 2 or 3 bytes
try(Reader reader = new FileReader("src/..")){in...
while(n=reader.read()!=-1)

~~try~~ FileOutputStream fos = new FileOutputStream("s.ser")
ObjectOutputStream oos = new ObjectOutputStream(fos)
oos.writeObject(student);
FileInputStream fis = new FIS(s.ser)
ObjectInputStream ois = new OIS(fis) oos.close()
Student stu = (Student) ois.readObject(); ois.close()
static Path createFile(Path path, FileAttribute<?>...attrs)
createDirectory/ies("./TempFile") java.nio.file.Files
static Path copy(Path from, Path to, CopyOption..options)
void delete ~~if~~ boolean exists(...
@override public FileVisitResult postVisitDirectory(Path, IO e)
directory sout return FileVisitResult.CONTINUE

```
Runnable cat = () -> Sout("Cat Thread")
Thread thread = new Thread(cat) thread.start();
Run public synchronized void method() -> this.intrinsicLock.lock()
try{ ... } finally{ this. ... unlock() }      wait() notifyAll()
public void withdraw(double amount) throws InterruEx {
    synchronized(this){
        while(balance < amount){ wait(); }
        balance -= amount; }
public void deposit ..
    synchronized(this){   notifyAll(); }
FixedThreadPool  ExecutorService es = Executors.newFixedThreadPool(3)
SingleThreadExecutor      .newSingleThreadExecutor()
CachedThreadPool     .newCachedThreadPool()   ☆
Future<T> submit(Callable<T> task)
Future<?> submit(Runnable task)   es.submit(()->sn"")
Future<V> get()  get(log timeout, TimeUnit unit)
cancel(boolean myInterrupt) boolean isCancelled() isDone()
Callable<String> task = ..   Future<String> future = execu.submit(task)
Set<Callable<String>> callables = new HashSet<>();
callables.add(()->"Task1")  2 3   String r = es.invokeAny(callables)
URLConnection public static void readByURLConnection(String url)
throws IOException{ URL u = new URL(url); URLConnection conn = u.
openConnection(); HttpURLConnection httpConn = (...) conn
int code = httpConn.getResponseCode()  String msg = httpConn.getRM
Sout . if ( code != HttpURLConnection.HTTP_OK){ return }
InputStream istream = httpConn.getIS   Scanner in = new S(istream)
while(in.hasNextLine()){ sout in.nextLine(); }
public static void readByHttpClient(String url) throws IOException
InterruptedException{ HttpClient client = HttpClient.newHC();
HttpRequest request = HR.newBuilder().uri(URI.create(url))
.GET().build(); HTTPResponse<String> response = client.send
(request, HTTPResponse.BodyHandlers.ofString());
String s = "https://api..."  URL url = new URL(s);
HTTPURLConnection conn = (HUC) url.openConnection();
conn.setRequestMethod("GET")  conn.connect  int responseCode
= conn.getRespoCode  String rmesg = conn.getRM  ContentEncoding
java.io.BufferedReader / IOException / InputStreamReader
java.net. ServerSocket / Socket
public class MutiUserServer{ psvm{ try{
  ServerSocket ss = new ServerSocket(8080)
  while true { Socket clientSocket = serS.accept()
  Thread clientThread = new Thread(()->handleClient(cS))
  CT.start() } catch (IOE e){ e.printStackTrace();
private static void handleClient(Socket clientSocket){
 try{ BufferedReader reader = new BR(new InputSR(cS.
  getIS())){ String msg while(msg = reader.readLine()) != null
  sout .} catch ..
java.io.IOException / PrintWriter  net.Socket  util.Scanner
public class Client{ psvm{ try{ Socket socket = new S("lh", 8080)
PrintWriter writer = new PrintWriter(socket.getOutputStream(), true)
while(true){ String msg = Scan  of writer.println(msg)
  writer.close  socket.close } catch
                                         box.getItems()
ChoiceBox<String> box = new ChoiceBox<String>();     .add()
box.getSelectionModel().selectedItemProperty().addListener
(ObservableValue<? extends String> observable, String oldValue,
String newValue) -> Sout (o -> n)
@Override  public void start(Stage primaryStage) throws Exce
    primiaryStage.setTitle   StackPane root = new SP
  Botton btn = new B   btn.setText   btn.setOnAction(new Ev
entHandler<ActionEvent>(){ @Override public void handle(
ActionEvent event){  root.getChildren().add(btn)
Scene scene = new Scene(root, 400, 200)  primaryStage.setScene(scene)
primaryStage.show()
playButton.setOnMouseClicked((event -> pathTransition.play()))
Class clz = bc.getClass() Field f = clz.getDeclaredField("balance")
f.setAccessible(true)   f.set(bc, 1000)
```

```
public Object invoke(Object obj, Object...args) throws
IllegalAccessException, IllegalArgumentEx, InvocationTargetE
Method m2 = Integer.class.getMethod("parseInt", String.class)
sout(m2.invoke(null, "12345").getClass().getName())
public T newInstance() throws InstantiationE, IllegalAccessE
Class cls = Student.class  Student s = (Student)cls.newInstance
Constructor con = clz.getConstructor(String.class, int.class)
Student std = (Student) constructor.newInstance(A, 15)
javax.servlet.ServletException / .WebServlet / .HttpServlet
.HttpServletRequest Response
@WebServlet(name = "hello", value = "/hello")
public class Hello extends HTTPServlet { private String msg
  public void init(){ msg = hello world
  public void doGet(HTTPServletRequest request,
HTTPServletResponse response) response.setContentType("text/html")
   PrintWriter out = response.getWriter().
   out.println("<html><body>"  public void destroy()
   @Configuration
   @ComponentScan(basePackages = "com.example")
   public class AppConfig{ @Bean public Engine engine()
   {return new Engine("V8"); }
@Controller
public class MyMvcController{ @RequestMapping(value = "/"
 method = RequestMethod.GET) public String prepareView
(Model model) model.addAttribute("msg", "Spring ...") return "myjsp"
@Entity @Table public class Student{ @Id @GeneratedValue
private Long id; private String name; private String email;
Thymeleaf server @Controller public class StudentController{
 private final StudentService ss. public StudentController
(SS ss){ this.ss = ss} @RequestMapping("/list")
public String getStudents(Model model){ model.addAttribute
("students", studentService.getStudents()), return "index"
@Service public Class StudentService{ private final
StudentRepository SR, @Autowired public StudentService
(SR sr){this. ...} public List<Student> getStudent(){
return SR.findAll(); public void addStudent Student x = new
SR.saveAll(List.of(x, y, z))        Entity/Modal Class  name   ID type
import com.example.mvcdemo.model.Student
org.springframework.data.jpa.repository.JpaRepository
public interface StudentRepository extends JpaR..<Student, Long>
 List<Student> findByEmailLike(String email)
@Transactional public void updateStudent          State
 Student s = SR.findById(sId).orElseThrow(()->new IllegalEx(...
@SpringBootApplication public class MVCDemoApplication{
public S psvm{ SpringApplication.run(MVCDApp.class, args)}
@Bean public CommandLineRunner CLR(S.S service){
 return args->{ service.addStudents(); }; }
  Private static final Logger = LoggerFactory.getLogger(SLDA.class)
 org.slf4j.Logger / LFactory
@SpringBootTest @AutoConfigureMockMvc
 @Class StudentRestControllerTest{ @Autowired private
  MockMvc mvc; @Test void getOneStud
  mvc.perform(get("/api/stu...sId), 1) andExpect
  {Status.isOk()).andExpect jsonPath("$.name").value(
  andDo(print())         is(stu.)...value) hasSize
.class  Platform Independence         value
Constant Pool 存放字面量.(final) 符号引用.   attributes
                                          field method
Magic versionConstantp.access_flag.this_class superclass interfaces
AV Java 15 jar on 16
field count ↑ fiels info 信息. 方法间
                                           验证 准备
Class File -> ClassLoader SubSystem          解析
              Loading -> Linking -> initializing
  Reachability
  Analysis            ↓    ↑
                   Thread1  Thread2        Native Method
  GC algorithm                             Library
                  Heap
                  Method Area
                  Runtime Constant Pool
     ↑↓                       ↑↓               ↑
Execution Engine
Interpreter JIT Compiler  Garbage Collector  Java Native Interface
```