Advanced Java Programming (4351603) - Winter 2024 Solution

Milav Dabgar

November 27, 2024

Question 1(a) [3 marks]

Describe JFC with its usage.

Solution

JFC (Java Foundation Classes) is a comprehensive GUI framework for building desktop applications in Java.

Table 1. JFC Components

Component	Description
Swing	Lightweight GUI components
AWT	Basic windowing toolkit
Java 2D	Advanced graphics and imaging
Accessibility	Support for assistive technologies

- **Primary Usage**: Creating rich desktop applications.
- **Key Advantage**: Platform independence and consistent look.

Mnemonic

"JFC = Java's Fantastic Components"

Question 1(b) [4 marks]

Explain Difference between AWT and Swing.

Solution

Table 2. AWT vs Swing

Feature	AWT	Swing
Components	Heavyweight (native)	Lightweight (pure Java)
Platform	Platform dependent	Platform independent
Look & Feel	Native OS look	Pluggable look and feel
Performance	Faster	Slightly slower

- **AWT Limitation**: Limited components, platform-specific appearance.
- **Swing Advantage**: Rich component set, customizable UI.

Mnemonic

"AWT = Always Weighs Too-much, Swing = Simply Works In New Generation"

Question 1(c) [7 marks]

List out various Event Listener. Explain anyone.

Solution

Event Listeners List:

Table 3. Common Event Listeners

Listener	Purpose
ActionListener	Button clicks, menu selections
MouseListener	Mouse events (click, press, release)
KeyListener	Keyboard input events
WindowListener	Window state changes
FocusListener	Component focus events
ItemListener	Checkbox/radio button changes

ActionListener Explanation:

- **Interface Method**: `actionPerformed(ActionEvent e)`
- **Usage**: Handles button clicks and menu actions
- **Implementation**: Anonymous class or lambda expression

```
button.addActionListener(e -> {
    System.out.println("Button clicked!");
});
```

Mnemonic

"AMKWFI Listeners = Action Mouse Key Window Focus Item"

Question 1(c OR) [7 marks]

List out various Layout Managers. Explain anyone.

Solution

Layout Managers List:

Table 4. Layout Managers

Layout Manager	Purpose
FlowLayout	Sequential component placement
BorderLayout	Five regions (North, South, East, West, Center)
GridLayout	Grid-based arrangement
CardLayout	Stack of components
BoxLayout	Single row or column
GridBagLayout	Complex grid with constraints

BorderLayout Explanation:
- **Default Layout**: For JFrame and JDialog.
- **Five Regions**: North, South, East, West, Center.
- **Resizing**: Center expands, others stay preferred size.

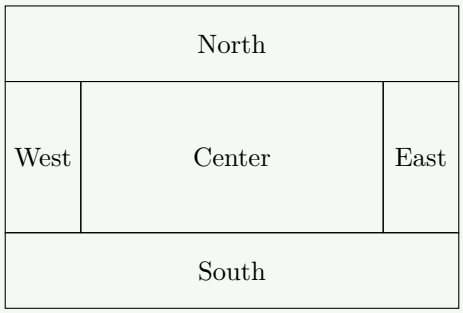

Figure 1. BorderLayout Regions

Mnemonic

"FBGCBG Layouts = Flow Border Grid Card Box GridBag"

Question 2(a) [3 marks]

List out and explain steps to connect database.

Solution

Database Connection Steps:

Step	Action
1. Load Driver	`Class.forName("driver.class")`
2. Create Connection	`DriverManager.getConnection()`
3. Create Statement	`connection.createStatement()`
4. Execute Query	`statement.executeQuery()`
5. Process Results	`resultSet.next()`
6. Close Resources	Close all connections

Mnemonic

"LCD EPR = Load Create Driver, Execute Process Results"

Question 2(b) [4 marks]

Explain 3-tier architecture with diagram.

Solution

3-tier architecture separates application into three logical layers for better maintainability.

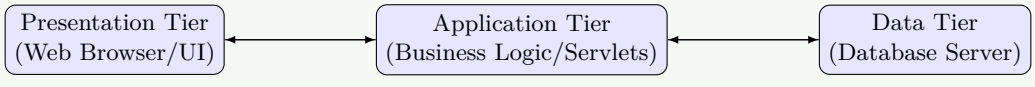

Figure 2. 3-Tier Architecture

Tier	Responsibility
Presentation	User interface and user interaction
Application	Business logic and processing
Data	Data storage and management

- **Advantage**: Better scalability and maintainability.
- **Example**: Web browser → Web server → Database.

Mnemonic

"PAD = Presentation Application Data"

Question 2(c) [7 marks]

Describe JDBC API with interfaces and classes.

Solution

JDBC API Components:

Table 5. JDBC Components

Type	Component	Purpose
Interface	Connection	Database connection
Interface	Statement	SQL execution
Interface	ResultSet	Query results
Interface	PreparedStatement	Precompiled SQL
Class	DriverManager	Driver management
Class	SQLException	Error handling

JDBC Architecture:

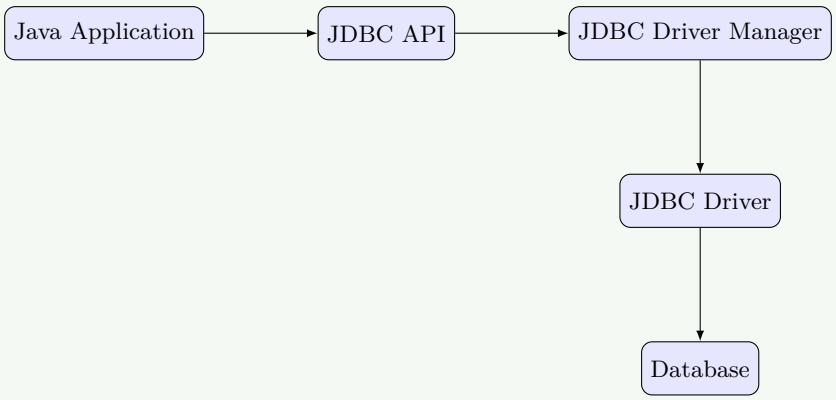

Figure 3. JDBC Architecture

Mnemonic

"CSRP Classes = Connection Statement ResultSet PreparedStatement"

Question 2(a OR) [3 marks]

List out advantages and disadvantages of JDBC.

Solution

Table 6. Advantages vs Disadvantages

Advantages	Disadvantages
Platform Independent	Performance Overhead
Standard API	Complex Configuration
Multiple Database Support	Limited ORM Features

Mnemonic

"PSM vs PCL = Platform Standard Multiple vs Performance Complex Limited"

Question 2(b OR) [4 marks]

Explain 2-tier architecture with diagram.

Solution

2-tier architecture directly connects client to database server.

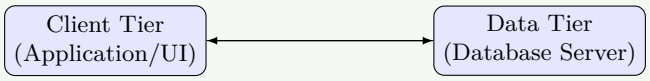

Figure 4. 2-Tier Architecture

- **Advantage**: Simple architecture, direct communication.
- **Disadvantage**: Limited scalability, tight coupling.
- **Example**: Desktop application connecting directly to database.

Mnemonic

"CD = Client Data (direct connection)"

Question 2(c OR) [7 marks]

List out JDBC driver types and Explain TYPE-4.

Solution

JDBC Driver Types:

Table 7. Driver Types

Type	Name	Description
Type-1	JDBC-ODBC Bridge	Uses ODBC driver
Type-2	Native-API Driver	Part Java, part native
Type-3	Network Protocol Driver	Pure Java, middleware
Type-4	Native Protocol Driver	Pure Java, direct

TYPE-4 Driver Explanation:

- **Pure Java**: Completely written in Java.
- **Direct Communication**: Directly communicates with database.
- **Platform Independent**: No native libraries required.
- **Best Performance**: Fastest among all types.
- **Examples**: MySQL Connector/J, PostgreSQL JDBC.

Figure 5. Type-4 Driver

Mnemonic

"ONNN Drivers = ODBC Native Network Native-pure"

Question 3(a) [3 marks]

Explain Application of servlet.

Solution

Servlet Applications:

Application	Usage
Web Forms	Process HTML form data
Database Operations	Connect and manipulate database
Session Management	Track user sessions
File Upload	Handle file uploads

Mnemonic

"WDSF = Web Database Session File"

Question 3(b) [4 marks]

Explain difference between Applet and Servlet.

Solution

Table 8. Applet vs Servlet

Feature	Applet	Servlet
Execution	Client-side (browser)	Server-side (web server)
Purpose	User interface	Request processing
Security	Restricted (sandbox)	Full server access
Performance	Limited by client	Server resources

Mnemonic

"Client vs Server = Applet vs Servlet"

Question 3(c) [7 marks]

Explain life cycle of a servlet in detail.

Solution

Servlet Life Cycle:

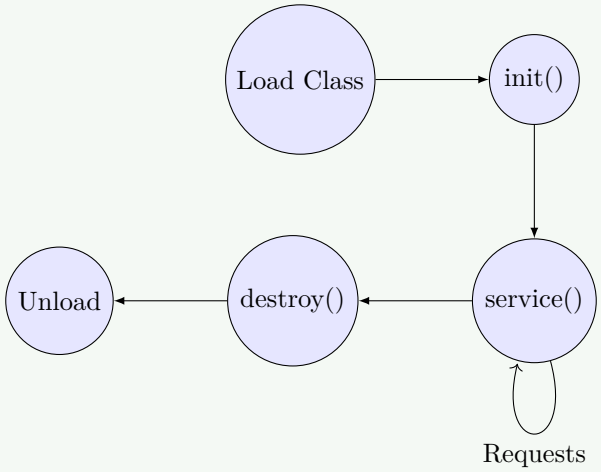

Figure 6. Servlet Life Cycle

Table 9. Life Cycle Phases

Phase	Method	Description
Loading	-	Web container loads servlet class
Initialization	`init()`	Called once, setup resources
Service	`service()`	Handles each request (doGet/doPost)
Destruction	`destroy()`	Cleanup before unloading

- **Thread Safety**: Multiple requests handled concurrently.
- **Single Instance**: One servlet instance handles all requests.

Mnemonic

"LISD = Load Init Service Destroy"

Question 3(a OR) [3 marks]

Explain web.xml file in servlet.

Solution

web.xml Purpose:

Element	Description
Deployment Descriptor	Configuration file for web application
Servlet Mapping	Maps URL patterns to servlets
Initialization	Servlet parameters and load order

- **Location**: WEB-INF directory
- **Format**: XML configuration file

Mnemonic

"DMI = Deployment Mapping Initialization"

Question 3(b OR) [4 marks]

List out and Explain feature of servlet.

Solution

Servlet Features:

Feature	Description
Platform Independent	Write once, run anywhere
Server-side	Executes on web server
Protocol Independent	Supports HTTP, FTP, etc.
Persistent	Stays in memory between requests
Secure	Built-in security features

Mnemonic

"PSPPS = Platform Server Protocol Persistent Secure"

Question 3(c OR) [7 marks]

Explain session tracking in servlet.

Solution

Session Tracking Methods:

Method	Description
Cookies	Small data stored in browser
URL Rewriting	Session ID in URL
Hidden Form Fields	Session data in forms
HttpSession	Server-side session object

HttpSession Logic Flow:

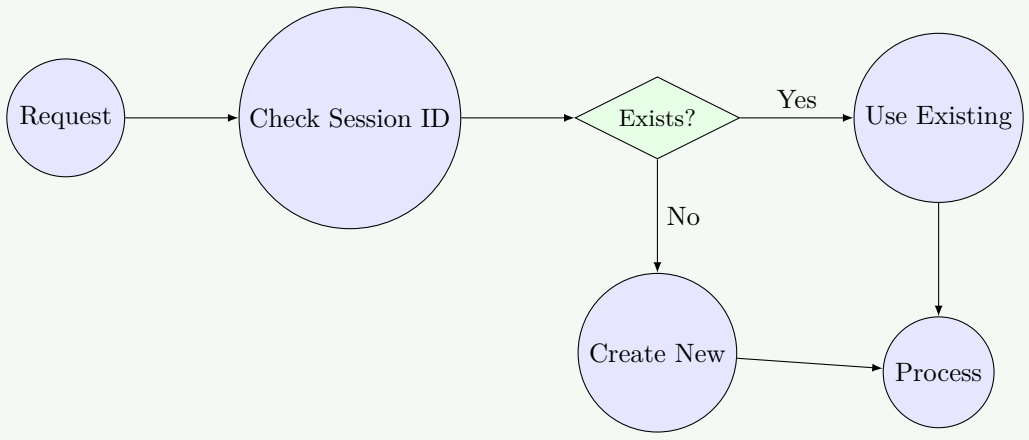

Figure 7. Session Tracking Logic

HttpSession Implementation:

```
HttpSession session = request.getSession();
session.setAttribute("user", username);
String user = (String) session.getAttribute("user");
```

Mnemonic

"CUHH = Cookies URL Hidden HttpSession"

Question 4(a) [3 marks]

Explain architecture of JSP with diagram.

Solution

JSP Architecture:

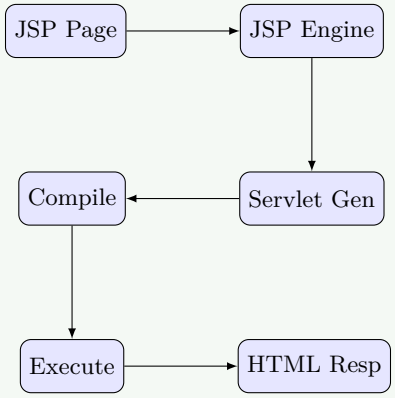

Figure 8. JSP Processing

1. **JSP Engine**: Translates JSP to servlet.
2. **Web Container**: Manages JSP lifecycle.
3. **Generated Servlet**: Actual execution unit.

Mnemonic

"JSP = Java Server Pages (Page to Servlet)"

Question 4(b) [4 marks]

Explain JSP scripting elements with example.

Solution

JSP Scripting Elements:

Table 10. Scripting Elements

Element	Syntax	Purpose
Scriptlet	`<% code %>`	Java code block
Expression	`<%= expression %>`	Output value
Declaration	`<%! declaration %>`	Variables/methods

Examples:

```
<%! int count = 0; %>                <!-- Declaration -->
<% count++; %>                       <!-- Scriptlet -->
<%= "Count: " + count %>             <!-- Expression -->
```

Mnemonic

"SED = Scriptlet Expression Declaration"

Question 4(c) [7 marks]

Explain JSP life cycle.

Solution

JSP Life Cycle Phases:

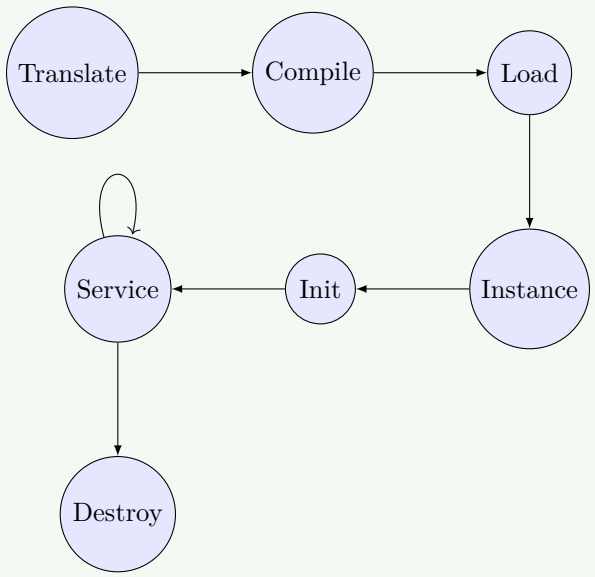

Figure 9. JSP Life Cycle

Phase	Description
Translation	JSP converted to servlet source
Compilation	Servlet source compiled to bytecode
Loading	Servlet class loaded by JVM
Instantiation	Servlet object created
Initialization	`jspInit()` method called
Request Processing	`_jspService()` handles requests
Destruction	`jspDestroy()` cleanup method

Mnemonic

"TCLIIRD = Translation Compilation Loading Instantiation Init Request Destroy"

Question 4(a OR) [3 marks]

Explain difference between JSP and Servlet.

Solution

Table 11. JSP vs Servlet

Feature	JSP	Servlet
Code Style	HTML with Java	Pure Java code
Development	Easier for UI	Better for logic
Compilation	Automatic	Manual
Modification	No recompilation needed	Requires recompilation

Mnemonic

"HTML vs Java = JSP vs Servlet"

Question 4(b OR) [4 marks]

List out and Explain advantage of JSP.

Solution

JSP Advantages:

Advantage	Description
Easy Development	HTML-like syntax with Java
Automatic Compilation	No manual compilation needed
Platform Independent	Runs on any Java-enabled server
Separation of Concerns	Design separated from logic
Reusable Components	Tag libraries and beans

Mnemonic

"EAPSR = Easy Automatic Platform Separation Reusable"

Question 4(c OR) [7 marks]

What is cookie? Explain how to Read and delete cookie using JSP page.

Solution

Cookie Overview: Cookie is a small piece of data stored on client's browser to maintain state.
Cookie Operations:

Operation	JSP Code
Create	`Cookie cookie = new Cookie("name", "value");`
Add	`response.addCookie(cookie);`
Read	`Cookie[] cookies = request.getCookies();`
Delete	`cookie.setMaxAge(0);`

Reading Cookie Example:

```
<%
Cookie[] cookies = request.getCookies();
if (cookies != null) {
    for (Cookie cookie : cookies) {
        if ("username".equals(cookie.getName())) {
            out.println("User: " + cookie.getValue());
        }
    }
}
%>
```

Deleting Cookie Example:

```
<%
Cookie cookie = new Cookie("username", "");
cookie.setMaxAge(0);
response.addCookie(cookie);
%>
```

Mnemonic

"CARD = Create Add Read Delete"

Question 5(a) [3 marks]

Explain importance of MVC architecture.

Solution

MVC Importance:

Benefit	Description
Separation of Concerns	Logic, presentation, data separated
Maintainability	Easy to modify individual components
Testability	Components can be tested independently

- **Code Organization**: Better structure and organization.
- **Team Development**: Multiple developers can work simultaneously.

Mnemonic

"SMT = Separation Maintainability Testability"

Question 5(b) [4 marks]

Explain Aspect oriented programming and dependency injection in brief.

Solution

Aspect Oriented Programming (AOP):

Concept	Description
Cross-cutting Concerns	Logging, security, transactions
Aspects	Modular units of cross-cutting functionality
Join Points	Points where aspects are applied

Dependency Injection (DI):

Concept	Description
Inversion of Control	Dependencies provided externally
Loose Coupling	Objects don't create dependencies
Configuration	Dependencies configured externally

Mnemonic

"AOP = Aspects Over Points, DI = Dependencies Injected"

Question 5(c) [7 marks]

Explain MVC architecture.

Solution

MVC Components:

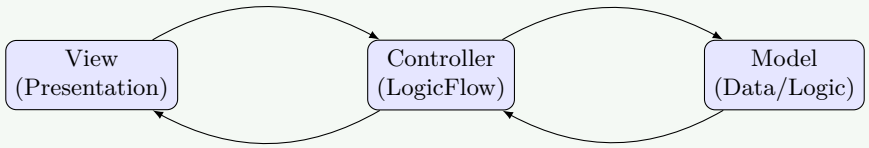

Figure 10. MVC Interaction

Component	Responsibility
Model	Business logic and data management
View	User interface and presentation
Controller	Request handling and flow control

MVC Flow:
1. **User Request** → Controller receives request.
2. **Controller** → Processes request, calls Model.
3. **Model** → Performs business logic, returns data.
4. **Controller** → Selects appropriate View.
5. **View** → Renders response to user.

Mnemonic

"MVC = Model View Controller (Business UI Control)"

Question 5(a OR) [3 marks]

Explain advantages of MVC architecture.

Solution

MVC Advantages:

Advantage	Description
Code Reusability	Components can be reused across applications
Parallel Development	Multiple developers work on different layers
Easy Testing	Each component tested independently
Maintenance	Changes in one layer don't affect others

Mnemonic

"CPEM = Code Parallel Easy Maintenance"

Question 5(b OR) [4 marks]

Explain difference between spring and spring boot.

Solution

Table 12. Spring vs Spring Boot

Feature	Spring	Spring Boot
Configuration	Manual XML/Java config	Auto-configuration
Setup Time	More setup required	Minimal setup
Embedded Server	External server needed	Built-in server
Dependencies	Manual dependency management	Starter dependencies

Mnemonic

"Manual vs Auto = Spring vs SpringBoot"

Question 5(c OR) [7 marks]

Explain architecture of Spring framework.

Solution

Spring Framework Architecture:

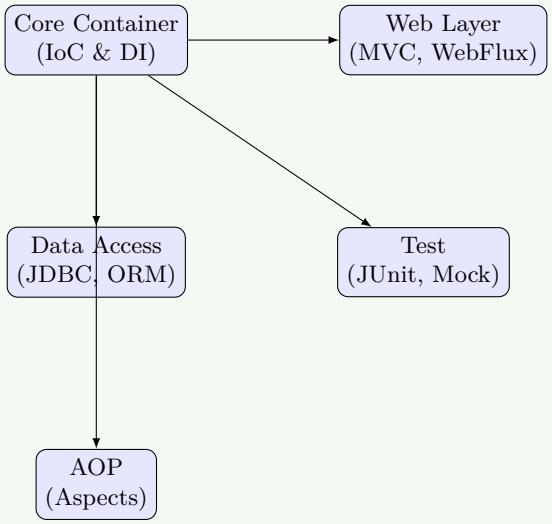

Figure 11. Spring Architecture

Spring Modules:

Module	Purpose
Core Container	IoC container, dependency injection
Data Access	JDBC, ORM, transaction management
Web	Web MVC, REST services
AOP	Aspect-oriented programming
Test	Testing support and mock objects

Mnemonic

"CDWAST = Core Data Web AOP Security Test"