

# Subject Name Solutions

4351603 – Winter 2024

Semester 1 Study Material

*Detailed Solutions and Explanations*

## 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.

Component	Description
<b>Swing</b>	Lightweight GUI components
<b>AWT</b>	Basic windowing toolkit
<b>Java 2D</b>	Advanced graphics and imaging
<b>Accessibility</b>	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

Feature	AWT	Swing
<b>Components</b>	Heavyweight (native)	Lightweight (pure Java)
<b>Platform</b>	Platform dependent	Platform independent
<b>Look &amp; Feel</b>	Native OS look	Pluggable look and feel
<b>Performance</b>	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:

Listener	Purpose
<b>ActionListener</b>	Button clicks, menu selections

<b>MouseListener</b>	Mouse events (click, press, release)
<b>KeyListener</b>	Keyboard input events
<b>WindowListener</b>	Window state changes
<b>FocusListener</b>	Component focus events
<b>ItemListener</b>	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:

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

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[North]
    B[West]
    C[Center]
    D[East]
    E[South]

    A --> B
    A --> C
    A --> D
    B --> C
    C --> D
    B --> E
    C --> E
    D --> E
{Highlighting}
{Shaded}
```

### 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	<code>Class.forName("driver.class")</code>
2. Create Connection	<code>DriverManager.getConnection()</code>
3. Create Statement	<code>connection.createStatement()</code>
4. Execute Query	<code>statement.executeQuery()</code>
5. Process Results	<code>resultSet.next()</code>
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.

### Mermaid Diagram (Code)

```

{Shaded}
{Highlighting}[]
graph LR
    A[Presentation Tier{br/{}Web Browser/UI} ]
    B[Application Tier{br/{}Business Logic/Servlets} ]
    C[Data Tier{br/{}Database Server}]

    A --> B
    B --> C
    {Highlighting}
    {Shaded}

```

Tier	Responsibility
<b>Presentation</b>	User interface and user interaction
<b>Application</b>	Business logic and processing
<b>Data</b>	Data storage and management

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

### Mnemonic

“PAD = Presentation Application Data”

## Question 2(c) [7 marks]

Describe JDBC API with interfaces and classes.

### Solution

#### JDBC API Components:

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

#### JDBC Architecture:

#### Mermaid Diagram (Code)

```

{Shaded}
{Highlighting}[]
graph LR
    A[Java Application] --> B[JDBC API]
    B --> C[JDBC Driver Manager]
    C --> D[JDBC Driver]
    D --> E[Database]
    {Highlighting}
    {Shaded}

```

- **Core Interfaces:** Connection, Statement, ResultSet, PreparedStatement
- **Key Classes:** DriverManager for connection management
- **Exception Handling:** SQLException for database errors

### Mnemonic

“CSRP Classes = Connection Statement ResultSet PreparedStatement”

### Question 2(a OR) [3 marks]

List out advantages and disadvantages of JDBC.

#### Solution

JDBC Advantages vs Disadvantages:

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

- **Benefits:** Write once, run anywhere with any database
- **Drawbacks:** Requires manual SQL and connection management

### 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.

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Client Tier{br/({}Application/UI] {}{-}{-}{} B[Data Tier{br/({}Database Server]]
{Highlighting}
{Shaded}
```

Tier	Responsibility
<b>Client</b>	User interface and business logic
<b>Server</b>	Data storage and management

- **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:

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

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Java Application] --{-}{-}{ B[Type{-}4 JDBC Driver{br/}{Pure Java}] --{-}{-}{ C[Database Server]}
{Highlighting}
{Shaded}
```

## Mnemonic

“ONNN Drivers = ODBC Native Network Native-pure”

## Question 3(a) [3 marks]

Explain Application of servlet.

## Solution

### Servlet Applications:

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

- **Primary Use:** Server-side Java programs for web applications
- **Common Tasks:** Request processing, response generation

## Mnemonic

“WDSF = Web Database Session File”

## Question 3(b) [4 marks]

Explain difference between Applet and Servlet.

### Solution

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

- **Applet:** Runs in web browser, limited capabilities
- **Servlet:** Runs on web server, full Java capabilities

### Mnemonic

“Client vs Server = Applet vs Servlet”

## Question 3(c) [7 marks]

Explain life cycle of a servlet in detail.

### Solution

Servlet Life Cycle:

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Servlet Class Loaded] --> B[init called]
    B --> C[service handles requests]
    C --> D[destroy called]
    D --> E[Servlet Unloaded]
{Highlighting}
{Shaded}
```

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

- **Thread Safety:** Multiple requests handled concurrently
- **Single Instance:** One servlet instance handles all requests
- **Container Managed:** Web container manages lifecycle

### Mnemonic

“LISD = Load Init Service Destroy”

## Question 3(a OR) [3 marks]

Explain web.xml file in servlet.

### Solution

web.xml Purpose:

Element	Description
<b>Deployment Descriptor</b>	Configuration file for web application
<b>Servlet Mapping</b>	Maps URL patterns to servlets
<b>Initialization</b>	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
<b>Platform Independent</b>	Write once, run anywhere
<b>Server-side</b>	Executes on web server
<b>Protocol Independent</b>	Supports HTTP, FTP, etc.
<b>Persistent</b>	Stays in memory between requests
<b>Secure</b>	Built-in security features

- **Performance:** Better than CGI scripts
- **Scalability:** Handles multiple requests efficiently

#### Mnemonic

“PSPPS = Platform Server Protocol Persistent Secure”

### Question 3(c OR) [7 marks]

Explain session tracking in servlet.

#### Solution

Session Tracking Methods:

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



### HttpSession Implementation:

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

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Client Request] --> B[Server checks Session ID]
    B --> C[Session exists?]
    C -- Yes --> D[Use existing session]
    C -- No --> E[Create new session]
    D --> F[Process Request]
    E --> F
{Highlighting}
{Shaded}
```

- **Purpose:** Maintain state across HTTP requests
- **HttpSession:** Most commonly used method

### Mnemonic

“CUHH = Cookies URL Hidden HttpSession”

## Question 4(a) [3 marks]

Explain architecture of JSP with diagram.

### Solution

#### JSP Architecture:

### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[JSP Page] --> B[JSP Engine/Container]
    B --> C[Servlet Code Generated]
    C --> D[Compiled to Bytecode]
    D --> E[Servlet Executed]
    E --> F[HTML Response]
{Highlighting}
{Shaded}
```

Component	Role
<b>JSP Engine</b>	Translates JSP to servlet
<b>Web Container</b>	Manages JSP lifecycle
<b>Generated Servlet</b>	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:

Element	Syntax	Purpose
<b>Scriptlet</b>	<code>&lt;% code %&gt;</code>	Java code block
<b>Expression</b>	<code>&lt;%= expression %&gt;</code>	Output value
<b>Declaration</b>	<code>&lt;#! declaration %&gt;</code>	Variables/methods

### Examples:

<code>{\%!} int count = 0; \%{}</code>	<code>{!{-}{-} Declaration {-}{-}}</code>
<code>{\%} count++; \%{}</code>	<code>{!{-}{-} Scriptlet {-}{-}}</code>
<code>{\%=} "Count: " + count \%{}</code>	<code>{!{-}{-} Expression {-}{-}}</code>

## Mnemonic

“SED = Scriptlet Expression Declaration”

## Question 4(c) [7 marks]

Explain JSP life cycle.

## Solution

### JSP Life Cycle Phases:

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[JSP Page Created] --> B[Translation to Servlet]
    B --> C[Servlet Compilation]
    C --> D[Class Loading]
    D --> E[Instantiation]
    E --> F[jspInit called]
    F --> G[_jspService handles requests]
    G --> H[jspDestroy called]
{Highlighting}
{Shaded}
```

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

- **Container Managed:** Web container handles entire lifecycle
- **Automatic:** Translation and compilation happen automatically

## Mnemonic

“TCLIRD = Translation Compilation Loading Instantiation Init Request Destroy”

### Question 4(a OR) [3 marks]

Explain difference between JSP and Servlet.

#### Solution

Feature	JSP	Servlet
<b>Code Style</b>	HTML with Java	Pure Java code
<b>Development</b>	Easier for UI	Better for logic
<b>Compilation</b>	Automatic	Manual
<b>Modification</b>	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
<b>Easy Development</b>	HTML-like syntax with Java
<b>Automatic Compilation</b>	No manual compilation needed
<b>Platform Independent</b>	Runs on any Java-enabled server
<b>Separation of Concerns</b>	Design separated from logic
<b>Reusable Components</b>	Tag libraries and beans

- **Developer Friendly:** Web designers can work with JSP easily
- **Maintenance:** Easier to modify than servlets

#### 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
<b>Create</b>	<code>Cookie cookie = new Cookie("name", "value");</code>
<b>Add</b>	<code>response.addCookie(cookie);</code>
<b>Read</b>	<code>Cookie[] cookies = request.getCookies();</code>
<b>Delete</b>	<code>cookie.setMaxAge(0);</code>

#### Reading Cookie Example:

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

#### Deleting Cookie Example:

```
{\%}  
Cookie cookie = new Cookie("username", "");  
cookie.setMaxAge(0);  
response.addCookie(cookie);  
%\n}
```

#### Mnemonic

“CARD = Create Add Read Delete”

### Question 5(a) [3 marks]

Explain importance of MVC architecture.

#### Solution

##### MVC Importance:

Benefit	Description
<b>Separation of Concerns</b>	Logic, presentation, data separated
<b>Maintainability</b>	Easy to modify individual components
<b>Testability</b>	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
<b>Cross-cutting Concerns</b>	Logging, security, transactions
<b>Aspects</b>	Modular units of cross-cutting functionality
<b>Join Points</b>	Points where aspects are applied

### Dependency Injection (DI):

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

### Mnemonic

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

### Question 5(c) [7 marks]

Explain MVC architecture.

### Solution

#### MVC Components:

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[View{br/{}Presentation Layer} ]
    B[Controller{br/{}Control Layer} ]
    C[Model{br/{}Business Logic}]

    A --> B
    B --> C
    C --> B
    B --> A
{Highlighting}
{Shaded}
```

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

#### MVC Flow:

1. **User Request** → *Controller receives request*
1. **Controller** → *Processes request, calls Model*
1. **Model** → *Performs business logic, returns data*
1. **Controller** → *Selects appropriate View*
1. **View** → *Renders response to user*

#### Advantages:

- **Maintainability:** Clear separation of responsibilities
- **Reusability:** Components can be reused
- **Testability:** Each layer can be tested independently

### Mnemonic

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

### Question 5(a OR) [3 marks]

Explain advantages of MVC architecture.

### Solution

#### MVC Advantages:

Advantage	Description
<b>Code Reusability</b>	Components can be reused across applications
<b>Parallel Development</b>	Multiple developers work on different layers
<b>Easy Testing</b>	Each component tested independently
<b>Maintenance</b>	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

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

- **Spring:** Comprehensive framework requiring configuration
- **Spring Boot:** Convention over configuration approach

### Mnemonic

"Manual vs Auto = Spring vs SpringBoot"

### Question 5(c OR) [7 marks]

Explain architecture of Spring framework.

### Solution

#### Spring Framework Architecture:

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph TD
    A[Core Container{br/{ }IoC \& DI} ]
    B[Data Access{br/{ }JDBC, ORM} ]
    C[Web Layer{br/{ }MVC, WebFlux}]
    D[AOP{br/{ }Aspect Oriented}]
    E[Test{br/{ }Testing Support}]

    A --> B
    A --> C
    A --> D
    A --> E
{Highlighting}
{Shaded}
```

### Spring Modules:

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

### Key Features:

- **IoC Container:** Manages object creation and dependencies
- **AOP Support:** Cross-cutting concerns handling
- **Transaction Management:** Declarative transaction support
- **MVC Framework:** Web application development

### Mnemonic

“CDWAST = Core Data Web AOP Security Test”