

Comprehensive Study Notes

Fundamentals to Java Programming

(Unit 5)

Class XI — Information Technology (Code 802)

 **Prepared by:**
Ashish Kumar Bhoi

Contents

1	NetBeans Integrated Development Environment (IDE)	2
1.1	Introduction to NetBeans IDE	2
1.2	Core Components of NetBeans IDE Window	2
1.3	Controls and Containers (Parent vs. Child Controls)	2
1.4	Workflow: Creating a Project and JFrame Form	3
1.5	Object-Oriented Nature of Swing Controls	3
2	Java Swing Components & API Reference	4
2.1	Summary of Standard Swing Controls	4
2.2	Detailed API Breakdown	4
2.2.1	1. JOptionPane Dialogs	4
2.2.2	2. ButtonGroup for Radio Buttons	5
2.2.3	3. JList Selection Modes	5
3	Java Programming Fundamentals	6
3.1	Object-Oriented Programming (OOP) Concepts	6
3.1.1	Access Specifiers & Visibility Modes	6
3.2	Variables & Naming Rules	6
3.3	Java Data Types	6
3.4	Operators in Java	7
3.5	Data Type Conversions & Wrapper Parsing	7
4	Control Structures & Practical Applications	9
4.1	Selection Statements	9
4.1.1	1. Simple if and if...else Statement	9
4.1.2	2. Nested if...else if...else Ladder	9
4.1.3	3. switch Statement	9
4.2	Complete Practical Application Implementations	10
4.2.1	Application 1: Simple Interest Calculator	10
4.2.2	Application 2: Simple Arithmetic Calculator	10
5	Comprehensive Question Bank	12
5.1	Multiple Choice Questions (MCQs)	12
5.2	Assertion-Reasoning Questions	13
5.3	One-Word / Brief Answer Questions	14
5.4	Differentiate Between Questions	15
5.5	Short Answer Questions	15
5.6	Long Answer & Code Writing Questions	16
5.7	Case-Study & Competency-Based Questions	17

1 NetBeans Integrated Development Environment (IDE)

1.1 Introduction to NetBeans IDE

An **Integrated Development Environment (IDE)** is a unified software suite that consolidates code editing, GUI building, compilation, debugging, and execution tools into a single user-friendly application.

NetBeans IDE provides an intuitive graphical environment for developing Java applications. Its built-in **GUI Builder** enables developers to construct front-end forms visually using drag-and-drop mechanics. The IDE automatically manages component placement, alignment, padding, and underlying Java code generation.

1.2 Core Components of NetBeans IDE Window

The NetBeans interface consists of eight fundamental workspace components:

1. **Title Bar:** Displays the active application project title, active form file name, and IDE version.
2. **Menu Bar:** Provides drop-down menus (File, Edit, View, Navigate, Source, Refactor, Run, Tools, Window, Help).
3. **Toolbars:** Shortcut icon buttons for rapid execution of frequent operations (Save All, Build Project, Run File Shift+F6, Undo, Redo).
4. **GUI Builder (Design Area):** The visual workspace where forms are created. It features two toggle views:
 - **Design View:** Visual drag-and-drop workspace for placing and positioning components.
 - **Source View:** Code editor displaying the generated and user-written Java source code.
5. **Palette:** Contains ready-to-use GUI components grouped under categories such as *Swing Controls*, *Swing Containers*, and *Swing Menus*.
6. **Inspector Window:** Displays a hierarchical tree view of all components placed on the form (containers, controls, menu items). It simplifies selecting hidden or overlapping components.
7. **Properties Window:** Allows developers to inspect and modify attributes (properties) of the currently highlighted visual component.
8. **Code Editor Window:** The environment where event handler methods and application logic are written.

1.3 Controls and Containers (Parent vs. Child Controls)

GUI visual elements are broadly divided into two structural categories:

Table 1: Classification of GUI Controls

Control Class	Definition	Characteristics & Examples
Parent / Container	A component that acts as a background container to hold other controls.	Examples: JFrame, JPanel. Moving or deleting a parent container automatically moves or deletes all its child controls.
Child Control	A control placed inside a container component to perform specific interaction tasks.	Examples: JButton, JTextField, JLabel, JRadioButton, JCheckBox.

1.4 Workflow: Creating a Project and JFrame Form

1. **Create New Project:** Select File > New Project (or click Toolbar icon). In Categories select **Java**, in Projects choose **Java Application**. Click **Next**.
2. **Configure Project Details:** Enter Project Name (e.g., SchoolApp). Uncheck "Create Main Class". Click **Finish**.
3. **Add a JFrame Form:** In the Projects window, right-click the project node, select New > JFrame Form. Specify Class Name (e.g., MainForm) and Package Name. Click **Finish**.
4. **Execute Application:** Press Shift + F6 (or select Run > Run File).

1.5 Object-Oriented Nature of Swing Controls

In Java Swing, every front-end visual component is an **Object** belonging to a predefined **Class** inside the `javax.swing` package.

- **Properties (Data Members):** Attributes storing appearance states (e.g., background, text, font, enabled).
- **Methods:** Functions invoked to perform operations on the control (e.g., `setText()`, `getText()`).
 - **Getters:** Retrieve values (start with `get` or `is`, e.g., `getText()`, `isSelected()`).
 - **Setters:** Change component properties (start with `set`, e.g., `setText()`, `setEnabled()`).
- **Events:** User actions performed on controls (e.g., `actionPerformed`, `mouseClicked`, `keyPressed`).

2 Java Swing Components & API Reference

2.1 Summary of Standard Swing Controls

Table 2: Swing Components & API Methods Summary

Component	Key Properties	Key Methods & Usage Syntax
JFrame	title, defaultCloseOperation	Top-level window container.
JButton	text, enabled, font, foreground	getText(), setText(String t), setEnabled(boolean b), setVisible(boolean b).
JTextField	text, editable, font, border	getText(), setText(String t), isEditable(), setEditable(boolean b). Single-line input.
JLabel	text, icon, horizontalAlignment	getText(), setText(String t). Displays read-only text or graphics.
JTextArea	rows, columns, lineWrap, wrapStyleWord	getText(), setText(String t), append(String str). Multi-line text entry/display.
JPasswordField	text, echoChar	getPassword() (returns char[]). Masked confidential entry.
JRadioButton	text, selected, buttonGroup	isSelected(), setSelected(boolean b). Mutually exclusive choice within a ButtonGroup.
JCheckBox	text, selected	isSelected(), setSelected(boolean b). Multi-selection toggle check box.
JComboBox	model, selectedIndex, editable	getSelectedItem(), setSelectedIndex(), setModel(). Drop-down choice list.
JList	model, selectedIndex, selectionMode	getSelectedValue(), setSelectedIndex(int i). Scrollable list component.

2.2 Detailed API Breakdown

2.2.1 1. JOptionPane Dialogs

JOptionPane provides standard pop-up dialog boxes for messages, inputs, and confirmations.

```
// Import statement required at top of file
import javax.swing.JOptionPane;

// 1. Message Dialog (Information Alert)
JOptionPane.showMessageDialog(this, "Welcome to Java Programming!");

// 2. Input Dialog (Prompt text box)
String name = JOptionPane.showInputDialog(this, "Enter Your Name:");
```

```
// 3. Confirm Dialog (Yes/No Question)
int response = JOptionPane.showConfirmDialog(this, "Are you sure you want
to exit?");
```

2.2.2 2. ButtonGroup for Radio Buttons

Radio buttons require a ButtonGroup instance to enforce mutual exclusivity (selecting one automatically deselects others).

```
// Associating radio buttons in NetBeans Properties or Code
ButtonGroup bg = new ButtonGroup();
bg.add(jRadioButton1);
bg.add(jRadioButton2);
```

2.2.3 3. JList Selection Modes

JList supports three distinct selection modes set via the selectionMode property:

- SINGLE_SELECTION: Allows selecting only one item at a time.
- SINGLE_INTERVAL_SELECTION: Allows selecting one contiguous block of items using the Shift key.
- MULTIPLE_INTERVAL_SELECTION: Allows selecting multiple non-contiguous items using the Ctrl key.

3 Java Programming Fundamentals

3.1 Object-Oriented Programming (OOP) Concepts

Object-Oriented Programming follows a bottom-up software design approach focused on data security and component reusability.

1. **Class:** A user-defined template or blueprint encapsulating data attributes and methods.
2. **Object:** An active instance of a class instantiated in computer memory.
3. **Encapsulation:** Wrapping data members and methods into a single unit (Class) while restricting direct internal access using visibility specifiers.
4. **Inheritance:** Mechanism allowing a sub-class (child) to inherit properties and methods from a super-class (parent).
5. **Polymorphism:** Ability of a method or operator to exhibit different behaviors based on context or signature.

3.1.1 Access Specifiers & Visibility Modes

- **private:** Accessible **only** within the defining class. (Recommended for data members).
- **public:** Accessible from any class across packages. (Recommended for methods).
- **protected:** Accessible within the same package and by child sub-classes.

3.2 Variables & Naming Rules

A **Variable** is a named location in memory used to temporarily store data values during program execution.

⚠ Java Variable Naming Rules

- Variable names are strictly **case-sensitive** (marks and MARKS are distinct).
- Must begin with a letter, an underscore (`_`), or a dollar sign (`$`).
- Cannot contain spaces or special characters (`#`, `@`, `!`, `%`, `&`).
- Cannot use reserved Java keywords (`int`, `class`, `public`, `static`, `if`).

3.3 Java Data Types

Data types specify the memory allocation, storage range, and valid operations for variables.

Table 3: Java Primitive Data Types Summary

Category	Type	Storage	Range	Example Declaration
Integer	byte	1 Byte (8 bits)	-128 to 127	byte rollNo = 15;
Integer	short	2 Bytes (16 bits)	-32,768 to 32,767	short age = 17;
Integer	int	4 Bytes (32 bits)	-2^{31} to $2^{31} - 1$	int salary = 45000;
Integer	long	8 Bytes (64 bits)	-2^{63} to $2^{63} - 1$	long dist = 9845000L;
Floating	float	4 Bytes (32 bits)	3.4×10^{-38} to 3.4×10^{38}	float avg = 88.5f;
Floating	double	8 Bytes (64 bits)	1.8×10^{-308} to 1.8×10^{308}	double principal = 50000.75;
Character	char	2 Bytes (16 bits)	Single Unicode character	char grade = 'A';
Boolean	boolean	1 bit	true or false	boolean isPassed = true;

String Type: String is a non-primitive **Reference Class** used to store sequences of characters enclosed in double quotes (e.g., `String name = "Ashish Bhoi";`).

3.4 Operators in Java

- **Assignment Operator (=):** Assigns right-hand expression value to left-hand variable (`int x = 10;`).
- **Arithmetic Operators:** + (Addition), - (Subtraction), * (Multiplication), / (Division), % (Modulus / Remainder).
- **Relational Operators:** == (Equal to), != (Not equal to), > (Greater than), < (Less than), >= (Greater or equal), <= (Less or equal).
- **Logical Operators:** && (Logical AND), || (Logical OR), ! (Logical NOT).

3.5 Data Type Conversions & Wrapper Parsing

Text retrieved from Swing components via `getText()` is always of type `String`. To execute numerical calculations, text strings must be parsed into numeric primitives, and computed results must be converted back to `String` for display.

Table 4: Type Conversion Methods & Wrapper Classes

Conversion Direction	Method Syntax	Description
String to Integer	<code>int n = Integer.parseInt(str);</code>	Parses numeric String into primitive int.
String to Double	<code>double d = Double.parseDouble(str);</code>	Parses decimal String into primitive double.
String to Byte	<code>byte b = Byte.parseByte(str);</code>	Parses byte String into primitive byte.
Integer to String	<code>String s = Integer.toString(n);</code>	Converts primitive int to String.
Double to String	<code>String s = Double.toString(d);</code>	Converts primitive double to String.
String Concatenation	<code>str1 + str2</code> OR <code>str1.concat(str2)</code>	Concatenates two String objects.

4 Control Structures & Practical Applications

4.1 Selection Statements

Selection statements control program flow by executing specific code blocks based on condition evaluation.

4.1.1 1. Simple if and if...else Statement

```
// Voting Eligibility Example
int age = Integer.parseInt(jTextField1.getText());

if (age >= 18) {
    JOptionPane.showMessageDialog(this, "You are Eligible to VOTE!");
} else {
    JOptionPane.showMessageDialog(this, "You are NOT Eligible to VOTE.");
}
```

4.1.2 2. Nested if...else if...else Ladder

Used when evaluating multiple non-mutually exclusive logical conditions.

```
// Week Day Finder using Radio Button checks
if (jRadioButton1.isSelected()) {
    jTextField1.setText("Monday");
} else if (jRadioButton2.isSelected()) {
    jTextField1.setText("Tuesday");
} else if (jRadioButton3.isSelected()) {
    jTextField1.setText("Wednesday");
} else if (jRadioButton4.isSelected()) {
    jTextField1.setText("Thursday");
} else if (jRadioButton5.isSelected()) {
    jTextField1.setText("Friday");
} else if (jRadioButton6.isSelected()) {
    jTextField1.setText("Saturday");
} else if (jRadioButton7.isSelected()) {
    jTextField1.setText("Sunday");
} else {
    jTextField1.setText("Day - Not Selected");
}
```

4.1.3 3. switch Statement

Evaluates an integer or character expression and transfers control to matching case labels.

```
// Discount Calculator based on JComboBox selected index
double billAmount = Double.parseDouble(jTextField1.getText());
```

```

double finalAmount = 0.0;

switch (jComboBox1.getSelectedIndex()) {
    case 0: // New Customer (0% Discount)
        finalAmount = billAmount;
        break;
    case 1: // Silver Customer (10% Discount)
        finalAmount = billAmount * 0.90;
        break;
    case 2: // Gold Customer (20% Discount)
        finalAmount = billAmount * 0.80;
        break;
    case 3: // Platinum Customer (30% Discount)
        finalAmount = billAmount * 0.70;
        break;
    default:
        finalAmount = billAmount;
}

jTextField2.setText(Double.toString(finalAmount));

```

4.2 Complete Practical Application Implementations

4.2.1 Application 1: Simple Interest Calculator

```

private void btnCalculateActionPerformed(java.awt.event.ActionEvent evt) {
    double principal = Double.parseDouble(txtPrincipal.getText());
    double rate = Double.parseDouble(txtRate.getText());
    byte time = Byte.parseByte(txtTime.getText());

    double simpleInterest = (principal * rate * time) / 100.0;

    txtInterest.setText(Double.toString(simpleInterest));
}

```

4.2.2 Application 2: Simple Arithmetic Calculator

```

// Addition Event Handler
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {
    double num1 = Double.parseDouble(txtNum1.getText());
    double num2 = Double.parseDouble(txtNum2.getText());
    double result = num1 + num2;
    txtResult.setText(Double.toString(result));
}

// Reset Fields Event Handler
private void btnResetActionPerformed(java.awt.event.ActionEvent evt) {
    txtNum1.setText("");
}

```

```
txtNum2.setText("");  
txtResult.setText("");  
}
```

5 Comprehensive Question Bank

5.1 Multiple Choice Questions (MCQs)

1. Which NetBeans window displays the hierarchical tree structure of components on a form?

- (A) Palette
- (B) Inspector Window
- (C) Properties Window
- (D) Code Editor

Answer: (B) Inspector Window

Explanation: The Inspector Window renders a hierarchical tree view of all container and child controls.

2. Which method retrieves the user-entered text from a JTextField?

- (A) setText()
- (B) getText()
- (C) readText()
- (D) parseText()

Answer: (B) getText()

Explanation: getText() extracts the String content from text components.

3. What is the memory size of the int data type in Java?

- (A) 1 Byte
- (B) 2 Bytes
- (C) 4 Bytes
- (D) 8 Bytes

Answer: (C) 4 Bytes

Explanation: Primitive int occupies 32 bits (4 bytes) of storage.

4. Which control is used when the user must select **exactly one** option from a set?

- (A) JCheckBox
- (B) JRadioButton
- (C) JTextArea
- (D) JLabel

Answer: (B) JRadioButton

Explanation: Radio buttons grouped in a ButtonGroup enforce single-option mutual exclusivity.

5. Which wrapper class method converts a String to a primitive integer?

- (A) Integer.parse()
- (B) Integer.parseInt()
- (C) String.toInt()
- (D) Integer.toString()

Answer: (B) Integer.parseInt()

Explanation: Integer.parseInt(String s) parses numeric strings to int.

6. What is the default echo character in a JPasswordField?

- (A) # (C) *
(B) \$ (D) @

Answer: (C) *

Explanation: By default, JPasswordField masks input characters with an asterisk (*).

7. Which invisible component is required to logically group JRadioButton controls?

- (A) JPanel (C) RadioGroup
(B) ButtonGroup (D) JList

Answer: (B) ButtonGroup

Explanation: A ButtonGroup coordinates radio buttons to ensure only one is selected at a time.

8. Which package must be imported to utilize JOptionPane dialogs?

- (A) java.awt.* (C) javax.swing.JOptionPane
(B) java.util.* (D) java.io.*

Answer: (C) javax.swing.JOptionPane

Explanation: JOptionPane resides inside the javax.swing package.

9. Which Swing component is best suited for accepting multi-line addresses?

- (A) JTextField (C) JLabel
(B) JTextArea (D) JPasswordField

Answer: (B) JTextArea

Explanation: JTextArea supports multi-line text input and display.

10. What is the return type of JRadioButton.isSelected()?

- (A) int (C) boolean
(B) String (D) char

Answer: (C) boolean

Explanation: isSelected() returns true if checked and false if unchecked.

5.2 Assertion-Reasoning Questions

1. **Assertion (A):** Deleting a JFrame container automatically deletes all JButton controls placed inside it.

Reason (R): A JFrame acts as a Parent Container control, and child controls inherit life cycle operations from their parent container.

Answer: Both (A) and (R) are true and (R) is the correct explanation of (A).

Explanation: Moving or deleting container components directly affects all contained child controls.

2. **Assertion (A):** The `switch` statement in Java can evaluate floating-point double variables.
Reason (R): The `switch` controlling expression must evaluate strictly to integer, character, byte, short, or String values.
Answer: (A) is false but (R) is true.
Explanation: `switch` does not accept float or double types due to precision comparison ambiguities.
3. **Assertion (A):** `Integer.parseInt(jTextField1.getText())` is necessary to perform mathematical operations on text field entries.
Reason (R): The `getText()` method returns data strictly as a String object.
Answer: Both (A) and (R) are true and (R) is the correct explanation of (A).
Explanation: Arithmetic operators cannot operate directly on String literals.
4. **Assertion (A):** Multiple `JRadioButton` controls on a form are mutually exclusive by default.
Reason (R): Mutual exclusivity requires manually associating the radio buttons with a `ButtonGroup` instance.
Answer: (A) is false but (R) is true.
Explanation: Radio buttons act independently until explicitly grouped using a `ButtonGroup`.

5.3 One-Word / Brief Answer Questions

1. **Question:** What does IDE stand for?
Answer: Integrated Development Environment.
2. **Question:** Name the NetBeans window containing draggable Swing controls.
Answer: Palette.
3. **Question:** Which method appends text to the end of a `JTextArea`?
Answer: `append()`.
4. **Question:** What key combination executes a file in NetBeans IDE?
Answer: Shift + F6.
5. **Question:** Name the property used to make a text field read-only.
Answer: `editable` (set to false).
6. **Question:** Which data type stores single characters enclosed in single quotes?
Answer: `char`.
7. **Question:** What keyword exits a `switch` statement case branch?
Answer: `break`.
8. **Question:** Which method converts an integer to its String representation?
Answer: `Integer.toString()`.
9. **Question:** Name the logical operator that reverses a boolean condition.
Answer: Logical NOT (!).
10. **Question:** What is the starting index number of items in a `JComboBox`?
Answer: 0.

5.4 Differentiate Between Questions

1. Differentiate between Parent (Container) Controls and Child Controls.

Feature	Parent Container Control	Child Control
Structural Role	Serves as a background container holding other components.	Placed inside a container to provide UI interaction.
Movement / Deletion	Moving/deleting parent moves/deletes all child controls.	Can be moved or deleted independently within container.
Examples	JFrame, JPanel.	JButton, JTextField, JLabel.

2. Differentiate between JTextField and JTextArea.

Feature	JTextField	JTextArea
Input Capacity	Accepts a single line of text entry.	Accepts multiple lines of text entry.
Scroll Bar Support	Does not support scroll bars.	Automatically incorporates vertical/horizontal scroll bars.
Key Methods	getText(), setText().	getText(), setText(), append().

3. Differentiate between JRadioButton and JCheckBox.

Feature	JRadioButton	JCheckBox
Selection Logic	Used for single selection from mutually exclusive choices.	Used for selecting multiple independent options.
Grouping Requirement	Requires ButtonGroup association for mutual exclusivity.	Operates independently without requiring button groups.
Best Use Case	Gender selection (Male/Female), Payment Mode.	Selecting favorite hobbies or food toppings.

4. Differentiate between if-else Ladder and switch Statement.

Feature	if-else Ladder	switch Statement
Condition Scope	Evaluates complex range conditions and boolean expressions.	Evaluates exact discrete equality matches (int, char).
Data Type Flexibility	Accepts all primitive data types and relational operators.	Restricted to integer, char, byte, short, and String types.
Execution Control	Evaluates expressions sequentially top-to-bottom.	Uses jump table logic with mandatory break statements.

5.5 Short Answer Questions

1. **Question:** What is NetBeans IDE? State two major benefits of using its GUI Builder.

Answer: NetBeans IDE is a Java software development environment. Its GUI builder allows visual drag-and-drop component placement and automatically generates layout code.

2. **Question:** Explain the role of the ButtonGroup component in NetBeans.
Answer: ButtonGroup is an invisible component that groups radio buttons together, ensuring mutual exclusivity so that selecting one button automatically deselects all others in the group.
3. **Question:** What is the purpose of wrapper parsing methods like Integer.parseInt()?
Answer: Integer.parseInt() converts text data retrieved from Swing controls as a String into a primitive int value so mathematical computations can be performed.
4. **Question:** Explain the difference between Design View and Source View in NetBeans.
Answer: Design View is a visual workspace for dragging and positioning components. Source View is the code editor where Java source code and event handlers are written.
5. **Question:** What is the function of the append() method in JTextArea?
Answer: The append(String str) method appends the specified text string to the end of the existing content in a JTextArea component.

5.6 Long Answer & Code Writing Questions

1. **Question:** Write complete Java GUI event code for a "Simple Interest Calculator" form containing txtPrincipal, txtRate, txtTime, and txtInterest. Calculate interest on button click and display result.

Answer:

```
private void btnCalculateActionPerformed(java.awt.event.ActionEvent evt)
{
    try {
        double p = Double.parseDouble(txtPrincipal.getText());
        double r = Double.parseDouble(txtRate.getText());
        double t = Double.parseDouble(txtTime.getText());

        double si = (p * r * t) / 100.0;

        txtInterest.setText(Double.toString(si));
    } catch (NumberFormatException e) {
        JOptionPane.showMessageDialog(this, "Please enter valid numeric inputs!");
    }
}
```

2. **Question:** Design a GUI code snippet that reads student marks from txtMarks and displays Grade in lblGrade according to rules: Marks $\geq 90 \rightarrow$ "A", $\geq 75 \rightarrow$ "B", $\geq 50 \rightarrow$ "C", else "Fail".

Answer:

```
private void btnGradeActionPerformed(java.awt.event.ActionEvent evt) {
    double marks = Double.parseDouble(txtMarks.getText());
    String grade = "";

    if (marks >= 90) {
        grade = "Grade A";
    } else if (marks >= 75) {
```

```

        grade = "Grade B";
    } else if (marks >= 50) {
        grade = "Grade C";
    } else {
        grade = "Fail";
    }

    lblGrade.setText(grade);
}

```

5.7 Case-Study & Competency-Based Questions

1. Case Study 1: Book Publisher Discount System

Scenario: A publishing house offers discounts based on customer category selected from a JComboBox:

- **Book Seller:** 25% Discount
- **School:** 20% Discount
- **Customer:** 5% Discount

Required Task: Write event code for the "Calculate Net Price" button taking gross amount from txtGross and setting final price in txtNet.

Solution Code:

```

private void btnCalculateNetActionPerformed(java.awt.event.ActionEvent
    evt) {
    double grossPrice = Double.parseDouble(txtGross.getText());
    double netPrice = 0.0;
    int choice = jComboBox1.getSelectedIndex();

    switch (choice) {
        case 0: // Book Seller (25% discount)
            netPrice = grossPrice * 0.75;
            break;
        case 1: // School (20% discount)
            netPrice = grossPrice * 0.80;
            break;
        case 2: // Customer (5% discount)
            netPrice = grossPrice * 0.95;
            break;
        default:
            netPrice = grossPrice;
    }

    txtNet.setText(Double.toString(netPrice));
}

```

2. Case Study 2: Employee Salary Calculator

Scenario: A company calculates employee salary components based on txtBasic input:

- If Basic $\geq$ 40000: DA = 35%, HRA = 37%
- If Basic $\geq$ 20000: DA = 25%, HRA = 32%
- If Basic $\geq$ 10000: DA = 25%, HRA = 30%

Solution Code:

```
private void btnCalcSalaryActionPerformed(java.awt.event.ActionEvent evt) {  
    double basic = Double.parseDouble(txtBasic.getText());  
    double da = 0.0, hra = 0.0;  
  
    if (basic >= 40000) {  
        da = 0.35 * basic;  
        hra = 0.37 * basic;  
    } else if (basic >= 20000) {  
        da = 0.25 * basic;  
        hra = 0.32 * basic;  
    } else if (basic >= 10000) {  
        da = 0.25 * basic;  
        hra = 0.30 * basic;  
    }  
  
    double grossSalary = basic + da + hra;  
    txtGrossSalary.setText(Double.toString(grossSalary));  
}
```