

UNDERGRADUATE LEARNING SERIES / UNIT I

JAVA PROGRAMMING FUNDAMENTALS

अवधारणाएँ, उदाहरण एवं अभ्यास

सरल Hindi-English व्याख्या के साथ

14 अध्यायों में Java की आधारभूत शिक्षा

CONCEPTS

PROGRAMS

MCQs

Hindi + English Edition
Overview of Java · Unit I

पुस्तक का परिचय

How to Use This Book

यह पुस्तक आपके Unit I syllabus के आधार पर Java का परिचय, development environment, source और JAR files, data types, operators, control statements, arrays, classes, constructors, nested classes, strings तथा packages को एक क्रम में प्रस्तुत करती है।

व्याख्या Hindi-English mix में है। Technical terms, Java syntax और commands English में रखी गई हैं। प्रत्येक अध्याय में explanations, examples, outputs, revision और अभ्यास दिए गए हैं।

पढ़ने का तरीका

पहले concept पढ़ें, फिर program स्वयं type करके चलाएँ। Output बदलने के लिए input values बदलें। MCQs हल करने के बाद answer key देखें। केवल code याद करने के स्थान पर execution की प्रक्रिया समझें।

Version और Program Conventions

अधिकांश examples traditional class-based Java syntax पर आधारित हैं। Java 9/11 की सुविधाओं का जहाँ उल्लेख है, वहीं version बताया गया है। Newer switch forms और अन्य advanced version-specific features इस introductory unit के दायरे से बाहर हैं।

Public class वाला program उसी class name की .java file में save करें। छोटे fragments किसी उचित method/class के भीतर उपयोग करें। अलग examples में समान helper class names हो सकते हैं; उन्हें अलग folders में रखें।

Symbols और Design

नीले boxes में code और commands, हल्के हरे tables में comparisons और पीले boxes में exam points दिए गए हैं। अध्याय 14 पूरे unit का revision, MCQ answer key और mini project प्रदान करता है।

विषय-सूची

Contents / Chapters 1-14

01. Java का परिचय एवं विकास	4
02. Java Development Environment	7
03. Program लिखना, Compile करना एवं चलाना	11
04. Data Types, Variables, Literals एवं Constants	15
05. Operators एवं Expressions	19
06. Conditional Statements	22
07. Looping Statements	25
08. Arrays	28
09. Classes, Objects एवं Methods	31
10. Constructors	34
11. Nested एवं Inner Classes	37
12. String Handling	40
13. Packages	43
14. Revision एवं Question Bank	46

UNIT I / CHAPTER 01

अध्याय 1: Java का परिचय एवं विकास

Introduction and Evolution of Java

1.1 सीखने के उद्देश्य

इस अध्याय में Java का अर्थ, उत्पत्ति, प्रमुख विशेषताएँ, उपयोग तथा platform independence समझेंगे। Source code, bytecode और JVM का संबंध Java learning की आधारशिला है।

1.2 Java क्या है?

Java एक high-level, class-based programming language है। इसमें program को classes और methods के माध्यम से व्यवस्थित किया जाता है। High-level का अर्थ है कि instructions programmer के लिए अपेक्षाकृत समझने योग्य रूप में लिखी जाती हैं। Java code को चलाने से पहले सामान्यतः compiler bytecode बनाता है; compatible JVM उसे execute करती है।

Java का उपयोग desktop applications, web backends, enterprise systems, APIs, Android development और scientific data processing में किया जाता है। Android का runtime standard Java SE JVM के समान नहीं है, यद्यपि Java language का वहाँ उपयोग होता है।

1.3 Genesis और History

Java का विकास Sun Microsystems में James Gosling और उनकी team द्वारा किया गया। प्रारंभिक language का नाम **Oak** था। उद्देश्य विभिन्न electronic devices के लिए portable software technology बनाना था। बाद में नाम Java हुआ और 1995 में सार्वजनिक रूप से प्रस्तुत की गई।

वर्ष	प्रमुख घटना
1991	प्रारंभिक project और Oak language का विकास
1995	Java की सार्वजनिक प्रस्तुति
1996	JDK 1.0 जारी हुआ
1998	Java 2 तथा J2SE 1.2
2004	Java 5: generics, annotations, autoboxing
2010	Oracle ने Sun Microsystems का अधिग्रहण किया
2014	Java 8: lambda expressions और Stream API

1.4 Java की प्रमुख विशेषताएँ

Simple: Explicit pointer arithmetic जैसी कुछ जटिल सुविधाएँ उपलब्ध नहीं हैं। फिर भी variables, control flow और object-oriented concepts सीखना आवश्यक है।

Object-oriented: Classes और objects से data तथा behavior व्यवस्थित होते हैं। Java में primitive types भी हैं, इसलिए हर value object नहीं है।

Platform independent: Bytecode किसी एक operating system का native executable नहीं है। Compatible runtime होने पर अलग platforms पर चल सकती है।

Portable: Primitive types के निर्धारित sizes और standard libraries portability में सहायता करते हैं। Native libraries या OS-specific paths पर निर्भर code में बदलाव लग सकता है।

Robust: Type checking, exception handling और array bounds checks कई errors पहचानने में सहायता करते हैं।

Security-related features: Bytecode verification तथा access control उपलब्ध हैं। ये हर application के सुरक्षित होने की गारंटी नहीं हैं।

Multithreaded: Program में अनेक execution threads बनाए जा सकते हैं। वास्तविक parallel execution hardware और scheduling पर निर्भर करता है।

Automatic memory management: Garbage Collector unreachable objects की memory reclaim कर सकता है। Files और connections को सही तरह close करना फिर भी आवश्यक है।

1.5 Platform Independence

Write Once, Run Anywhere (WORA) का आशय है कि compiled bytecode compatible Java environments में चल सकती है। JVM implementation स्वयं operating system और processor के अनुसार बनी होती है।

चरण	File या Component	भूमिका
Source	Hello.java	Programmer का Java code
Compilation	javac	Source से bytecode
Bytecode	Hello.class	JVM instructions और metadata
Execution	Compatible JVM	Program चलाना

Exam Point: Bytecode platform independent हो सकती है; JVM implementation platform dependent होती है। नया class-file version पुराने JVM पर हमेशा नहीं चलता।

1.6 पहला Program

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Namaste! Welcome to Java.");
    }
}
```

Output: Namaste! Welcome to Java.

Code को Hello.java में save करें। `javac Hello.java` से compile करें और `java Hello` से चलाएँ। Java case-sensitive है; Hello और hello अलग नाम हैं।

1.7 Quick Revision एवं अभ्यास

- Java का प्रारंभिक नाम Oak था।
- James Gosling प्रमुख developer थे।
- Source extension .java और compiled class extension .class है।
- JIT runtime पर कुछ code को native machine code में compile कर सकता है।
- Java केवल interpreted language नहीं है।

लघु प्रश्न: Java क्या है? WORA का अर्थ लिखिए। Bytecode क्या है?

दीर्घ प्रश्न: Java की विशेषताएँ समझाइए। Platform independence उदाहरण सहित समझाइए।

MCQs: 1. Java का प्रारंभिक नाम: A. Pine B. Oak C. BASIC D. Pascal 2. Bytecode execute करने वाला component: A. Editor B. JVM C. javac D. javadoc 3. Java की source file: A. .java B. .class C. .jar D. .exe

उत्तर: 1-B, 2-B, 3-A.

UNIT I / CHAPTER 02

अध्याय 2: Java Development Environment

JVM, JRE, JDK एवं Development Tools

2.1 सीखने के उद्देश्य

Java development के लिए editor, compiler, runtime और debugging tools की भूमिका समझेंगे। JVM architecture तथा environment variables का basic परिचय प्राप्त करेंगे।

2.2 Development Environment

Java Development Environment उन tools का समूह है जिनसे code लिखते, compile करते, चलाते और debug करते हैं। Text editor या IDE source code तैयार करता है। JDK compiler और development tools देता है। Runtime execution की व्यवस्था करता है।

घटक	काम
Editor / IDE	Source लिखना और edit करना
Compiler	Type और syntax checking; bytecode तैयार करना
Runtime	Compiled application चलाना
Debugger	Execution और variable values की जाँच
Libraries	Reusable classes और methods

2.3 JVM

Java Virtual Machine एक abstract computing machine है जो class-file instructions execute करती है। इसके software implementations विभिन्न operating systems और processors के लिए उपलब्ध होते हैं।

इसके मुख्य भाग: class loading system, runtime data areas, execution engine तथा native code interface/support हैं।

Loading: Class की binary representation प्राप्त करके runtime representation बनाना।

Linking: Verification, preparation और resolution शामिल होते हैं। Verification bytecode के नियम जाँचती है। Preparation static fields को default storage values देती है। Resolution symbolic references को संबंधित runtime entities से जोड़ता है; कुछ resolution बाद में हो सकती है।

Initialization: Static field initializers और static blocks execute होते हैं। Loading और initialization एक घटना नहीं हैं।

2.4 Runtime Data Areas

Area	क्या रखता है?	Sharing
Heap	Objects और arrays	Shared
Method Area	Class structures, code, runtime constant pools	Shared
JVM Stack	Method invocation frames	प्रत्येक thread का अलग
PC Register	Current instruction की position	प्रत्येक thread का अलग
Native Method Stack	Native execution support, जहाँ लागू	Thread-related

Method frame में local variables तथा operand stack होते हैं। Method पूरा होने पर frame हटता है। JVM specification के logical areas और किसी implementation की physical memory व्यवस्था पूरी तरह समान होना आवश्यक नहीं है।

2.5 Interpreter, JIT और Garbage Collector

Interpreter bytecode instructions execute करता है। कई JVM implementations में JIT frequently executed code को native machine instructions में compile करता है। हर JVM के लिए दोनों techniques अनिवार्य नहीं हैं।

Garbage Collector unreachable objects की memory reclaim कर सकता है। `System.gc()` केवल अनुरोध है; तत्काल collection की गारंटी नहीं। किसी reference को null करने से object तुरंत delete नहीं होता।

2.6 JRE और JDK

JRE (Java Runtime Environment) application चलाने के लिए JVM, runtime libraries और support प्रदान करता है। **JDK (Java Development Kit)** compiler, development tools और runtime components प्रदान करता है। Modern distributions में अलग standalone JRE या jre folder हमेशा नहीं होता।

आधार	JVM	JRE	JDK
उद्देश्य	Bytecode execution	Application चलाना	Application विकसित करना
javac compiler	नहीं	सामान्यतः नहीं	हाँ
उपयोग	Execution component	Runtime environment	Programmer toolkit

2.7 प्रमुख JDK Tools

Tool	कार्य
javac	Source compilation
java	Application launcher

Tool	कार्य
jar	Java archives
javadoc	API documentation
javap	Class information / disassembly
jdb	Debugging
jshell	Interactive snippets; Java 9 से
jlink	Custom runtime image; Java 9 से

2.8 Setup और Environment Variables

JDK install करने के बाद terminal में `java -version` तथा `javac -version` चलाएँ। अनेक installations होने पर दोनों commands की location भी जाँचें।

PATH executable commands खोजने वाली directories की सूची है। JDK की bin directory उपलब्ध होने पर `javac` किसी folder से call किया जा सकता है। **JAVA_HOME** सामान्यतः JDK root को बताता है, bin folder को नहीं।

Classpath application classes और JAR files की locations बताता है।

```
JAVA_HOME = C:\Java\jdk
PATH entry = %JAVA_HOME%\bin
```

```
javac Hello.java
java -cp . Hello
```

यहाँ dot current directory है। Beginner programs के लिए global CLASSPATH set करना सामान्यतः आवश्यक नहीं है।

2.9 IDE एवं Common Errors

IntelliJ IDEA, Eclipse और Apache NetBeans IDE के उदाहरण हैं। IDE syntax highlighting, completion, debugging और project management देती है।

Error	सम्भावित कारण
javac command नहीं मिली	JDK/PATH की समस्या
file not found	गलत directory या filename
Main class नहीं मिली	Classpath या class name गलत
UnsupportedClassVersionError	Runtime bytecode के version से पुराना

अभ्यास: Installed versions जाँचें। Hello program command-line और IDE दोनों से चलाएँ। JVM, JRE और JDK की तुलना लिखें।

MCQs: 1. Development toolkit: A. JVM B. JDK C. Class file D. PATH 2. Object memory का logical area: A. Heap B. PC C. Source D. Manifest 3. PATH खोजता है: A. केवल classes B. Executable commands C. Objects D. Comments

उत्तर: 1-B, 2-A, 3-B.

UNIT I / CHAPTER 03

अध्याय 3: Program लिखना, Compile करना एवं चलाना

Source Files, Bytecode, Execution एवं JAR Files

3.1 सीखने के उद्देश्य

Source structure, main method, output statements, comments, command-line arguments, errors और executable JAR समझेंगे।

3.2 Development Process

Problem analysis से अपेक्षित input और output तय करें। फिर source code लिखें, compile करें, execute करें और expected result से तुलना करें। केवल compilation सफल होना logic सही होने की गारंटी नहीं है।

3.3 Program Structure

```
public class CollegeInfo {
    public static void main(String[] args) {
        System.out.println("Indira Gandhi Govt P G College");
        System.out.println("Vaishali Nagar, Bhilai");
    }
}
```

इसे CollegeInfo.java में save करें। File-based workflow में public top-level class और filename समान होते हैं। Curly braces class तथा method blocks को सीमित करती हैं। कई statements semicolon से समाप्त होते हैं; हर line semicolon से समाप्त नहीं होती।

3.4 Traditional main Method

भाग	भूमिका
public	Public access
static	Object बनाए बिना class-related invocation
void	Return value नहीं
main	Traditional entry-point name
String[]	Arguments का array
args	Parameter name; बदल सकते हैं

इस पुस्तक में traditional `public static void main(String[] args)` pattern उपयोग किया गया है।

3.5 Comments और Output

```
// Single-line comment
/* Multi-line comment */
/** Documentation comment */
```

Comments सामान्य instructions की तरह execute नहीं होते। Documentation comments javadoc में उपयोगी हैं।

Method	व्यवहार
<code>print()</code>	अपने आप line समाप्त नहीं करता
<code>println()</code>	Output के बाद line समाप्त करता है
<code>printf()</code>	Format के अनुसार output

```
System.out.print("Hello ");
System.out.println("Students");
System.out.printf("Marks: %d\n", 85);
```

Output: पहली line Hello Students; दूसरी line Marks: 85। `%n` platform line separator है।

3.6 Compilation और Execution

```
javac CollegeInfo.java
java CollegeInfo
```

Compiler class files बनाता है। एक source file से अनेक class files बन सकती हैं। Traditional class-launch command में `.class` extension नहीं लिखते। Java 11 से supported single-file source mode में `java CollegeInfo.java` भी संभव है; यह अलग launch mode है।

3.7 Command-Line Arguments

```
public class Greeting {
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please provide your name.");
            return;
        }
        System.out.println("Welcome, " + args[0]);
    }
}
```

```
javac Greeting.java
java Greeting "Suresh Thakur"
```

Output: Welcome, Suresh Thakur

Arguments strings होते हैं। args.length उनकी संख्या है और args[0] पहला argument है। बिना length जाँचे nonexistent argument पढ़ना exception दे सकता है।

3.8 Errors

प्रकार	कब सामने आती है?	उदाहरण
Compilation	Compile time	Missing semicolon, incompatible types
Runtime	Execution	Integer division by zero
Logical	Result checking	Area में multiplication की जगह addition

```
int length = 10;
int width = 5;
int area = length * width; // Correct area: 50
```

3.9 JAR और Manifest

JAR = Java Archive. यह ZIP-based archive है, जिसमें classes और resources रख सकते हैं। हर JAR executable नहीं होती। Manifest सामान्यतः META-INF/MANIFEST.MF में होती है। Executable JAR की main-class metadata entry point बताती है।

```
javac Hello.java
jar cfe HelloApp.jar Hello Hello.class
java -jar HelloApp.jar
jar tf HelloApp.jar
```

c create, f filename और e entry-point class है। External dependencies archive में अपने आप शामिल नहीं होतीं।

3.10 Output Folder

```
javac -d out Hello.java
java -cp out Hello
```

Compiled classes को source से अलग रख सकते हैं।

Common Mistake: Hello.java.txt नाम से save करना, source बदलने के बाद recompile न करना या गलत classpath देना।

अभ्यास: Name और college display करें। Command-line greeting बनाएँ। Executable JAR तैयार करें। Source, bytecode और native code की तुलना लिखें।

MCQs: 1. JAR का पूरा नाम: A. Java Archive B. Java Array Runtime C. Joint API D. Java Record 2. First argument: A. args[1] B. args[0] C. args.first D. args.length 3. Traditional run command: A. java Hello B. java Hello.class C. javac Hello D. run Hello

उत्तर: 1-A, 2-B, 3-A.

UNIT I / CHAPTER 04

अध्याय 4: Data Types, Variables, Literals एवं Constants

Values, Scope एवं Type Conversion

4.1 सीखने के उद्देश्य

Primitive/reference types, variables, literal forms, default values, constants, widening और narrowing समझेंगे।

4.2 Data Type

Data type allowed values और operations निर्धारित करता है। Java statically typed language है। `int marks = 80;` में `marks` का type `int` है; उसे सीधे String assign नहीं कर सकते।

Type	आकार	Range / Nature
byte	8 bits	-128 से 127
short	16 bits	-32768 से 32767
int	32 bits	-2^{31} से $2^{31}-1$
long	64 bits	-2^{63} से $2^{63}-1$
float	32 bits	लगभग 6-7 significant decimal digits
double	64 bits	लगभग 15-16 significant decimal digits
char	16 bits	Unsigned UTF-16 code unit, 0-65535
boolean	Fixed size language में specified नहीं	true या false

Signed n-bit range: $-2^{(n-1)}$ से $2^{(n-1)}-1$ । String, arrays और class/interface types reference types हैं।

4.3 Literals

```
int marks = 85;
long population = 30000000000L;
float rate = 5.5F;
double percentage = 86.75;
char grade = 'A';
boolean passed = true;
String name = "Amit";
```

Floating literal का default type double है। Character single quotes और String double quotes में है। char हर Unicode character को अकेले represent नहीं करता; कई emoji को दो UTF-16 code units चाहिए।

4.4 Number-System Literals

System	Example	Decimal value
Decimal	25	25
Binary	0b11001	25
Octal	031	25
Hexadecimal	0x19	25

010 decimal 8 है। 08 invalid octal literal है। Numeric literals में readable separators: 1_00_000। Underscore को literal के अंत, decimal point के adjacent या suffix के पहले नहीं रखें। Scientific notation: 1.5e3 का मान 1500.0 है।

4.5 Variables और Identifiers

Declaration: `int marks;` Initialization: `int marks = 80;` Assignment: `marks = 90;` Identifier digit से शुरू नहीं होता, space नहीं रखता और keyword नहीं हो सकता। Java case-sensitive है। अकेला underscore modern Java में identifier नहीं है।

Variable	कहाँ घोषित?	Default value?
Local	Method/block	नहीं
Instance	Class में non-static field	हाँ
Static	Class में static field	हाँ

Fields और array elements के defaults: numeric zero, boolean false, char U+0000 और references null। Local को पढ़ने से पहले definite assignment चाहिए। Scope नाम की visibility है; lifetime अलग अवधारणा है।

4.6 Constants

```
final int MAX_MARKS = 100;
```

इसे reassign नहीं कर सकते। हर final variable compile-time constant नहीं है। Constant variable final primitive/String है जो constant expression से initialized हो।

```
final int[] values = {10, 20};
values[0] = 99; // Valid: elements mutable
```

Final reference object को immutable नहीं बनाता।

4.7 Conversion

Widening examples: byte से short/int/long/float/double; char से int/long/float/double। Widening में int से float जैसी conversions precision खो सकती हैं।

```
int marks = 85;
double value = marks; // 85.0

double percentage = 86.75;
int whole = (int) percentage; // 86
```

Narrowing cast fractional part truncate कर सकती है। Integer narrowing low-order bits रखती है: (byte)130 का result -126 है।

4.8 Promotion और Parsing

```
byte a = 10, b = 20;
int sum = a + b; // Addition promoted to int
```

byte result = a + b; invalid है। लेकिन range में constant assignment byte x = 100; valid है।

```
double wrong = 5 / 2;           // 2.0
double correct = 5.0 / 2;       // 2.5
int marks = Integer.parseInt("85");
```

Parsing numerical text पढ़ती है; String से int का cast नहीं है। Invalid numerical text NumberFormatException दे सकती है।

4.9 Complete Program

```
public class StudentData {
    public static void main(String[] args) {
        int obtained = 425;
        final int MAX_MARKS = 500;
        double percentage = (double) obtained / MAX_MARKS * 100;
        System.out.println("Percentage: " + percentage);
        System.out.println("Passed: " + (percentage >= 40));
    }
}
```

Output: Percentage: 85.0 और Passed: true।

अभ्यास: सभी primitive types उपयोग करें; five subjects की percentage निकालें; narrowing के examples समझाएँ।

MCQs: 1. Primitive types की संख्या: A. 6 B. 7 C. 8 D. 9 2. String: A. Primitive B. Reference type C. Keyword D. Operator 3. (int)-12.9: A. -13 B. -12 C. 12 D. Error 4. Local को बिना assignment पढ़ना: A. 0 B. null C. Compile error D. हमेशा runtime error

उत्तर: 1-C, 2-B, 3-B, 4-C.

UNIT I / CHAPTER 05

अध्याय 5: Operators एवं Expressions

गणना, तुलना, Logic एवं Bit Operations

5.1 मूल अवधारणाएँ

Operator operation का symbol है। Operand उस operation की value है। Expression evaluation से result देता है। $a + b$ में $+$ operator और a, b operands हैं। Unary में एक, binary में दो और ternary conditional में तीन operands हैं।

5.2 Arithmetic Operators

Operator	कार्य	Example
+	Addition / string concatenation	$17 + 5 = 22$
-	Subtraction	$17 - 5 = 12$
*	Multiplication	$17 * 5 = 85$
/	Division	$17 / 5 = 3$, integer operands
%	Remainder	$17 \% 5 = 2$

Integer division zero की दिशा में truncate होती है। $-17/5 = -3$ और $-17\%5 = -2$ । Nonzero remainder का sign dividend के अनुसार है। Odd check के लिए $\text{number} \% 2 \neq 0$ negative values पर भी सही है।

Integer zero divisor ArithmeticException देता है। Floating examples: $10.0/0.0$ Infinity और $0.0/0.0$ NaN हैं।

5.3 Concatenation

```
System.out.println(10 + 20 + "Java"); // 30Java
System.out.println("Java" + 10 + 20); // Java1020
System.out.println("Java" + (10 + 20)); // Java30
```

Parentheses numerical addition पहले कराती हैं।

5.4 Increment और Decrement

Prefix $++x$ increment करके updated value देता है। Postfix $x++$ पुरानी value देता है लेकिन variable increment हो जाता है।

```
int x = 5;
int result = x++ + ++x;
```

```
System.out.println(result); // 12
System.out.println(x);      // 7
```

`x = x++`; में पुराने result का assignment increment को overwrite कर देता है। वास्तविक code में ऐसे complex side effects से बचकर सरल statements रखें।

5.5 Comparison और Logic

Category	Operators	Result
Relational	<, >, <=, >=	boolean
Equality	==, !=	boolean
Logical	&&, , !	boolean

= assignment है; == equality है। References पर == identity जाँचता है। String content के लिए equals उपयोग करें।

A	B	A && B	A B
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

5.6 Short-Circuit

```
int divisor = 0;
boolean valid = divisor != 0 && 10 / divisor > 1;
```

पहला operand false है, इसलिए division skip होती है। || में पहला operand true हो, तो second skip होता है। Boolean &, | और ^ दोनों operands evaluate करते हैं; ये short-circuit नहीं हैं।

5.7 Bitwise एवं Shift

Expression	Result
12 & 10	8
12 10	14
12 ^ 10	6

Expression	Result
~12	-13
5 << 2	20
20 >> 2	5
-5 >> 1	-3

XOR में अलग bits पर 1 मिलता है। Complement के लिए $\sim x = -x - 1$ । $\wedge$ power नहीं है; exponentiation के लिए Math.pow उपयोग करें। >> sign extension करता है और >>> zeros भरता है। Int shift distance के low 5 bits और long के low 6 bits उपयोग होते हैं; $1 \ll 32 = 1$ ।

5.8 Assignment और Conditional

```
int x = 10;
x += 5; // 15
String result = x >= 10 ? "Large" : "Small";
```

Compound assignment implicit conversion शामिल करती है। byte x=10; x+=1; valid है, लेकिन x=x+1; cast के बिना invalid है। Conditional operator केवल selected result expression evaluate करता है। null instanceof String false है।

5.9 Precedence एवं Evaluation

High से low मुख्य क्रम: unary, multiplication/division/remainder, addition/subtraction, shift, relational, equality, bitwise AND/XOR/OR, logical AND/OR, conditional, assignment। Parentheses grouping स्पष्ट करती हैं।

Precedence grouping बताती है; associativity समान precedence की grouping बताती है। Java operands सामान्यतः left-to-right evaluate करती है। Short-circuit rules applicable operands skip कर सकते हैं।

5.10 Overflow

```
int a = 50_000, b = 50_000;
long product = (long) a * b;
System.out.println(product); // 2500000000
```

Cast के बिना multiplication int में overflow कर सकती है, भले destination long हो। Ordinary integer arithmetic overflow पर automatic exception नहीं देती।

अभ्यास: Bill और discount calculation, maximum of two, bitwise outputs और increment traces।

MCQs: 1. $\wedge$ का अर्थ: A. Power B. XOR C. Division D. Assignment 2. ~5: A. 5 B. -5 C. -6 D. 6 3. $4 \ll 2$: A. 1 B. 8 C. 16 D. 32 4. Short-circuit AND: A. & B. && C. $\wedge$ D. ~

उत्तर: 1-B, 2-C, 3-C, 4-B.

UNIT I / CHAPTER 06

अध्याय 6: Conditional Statements

निर्णय लेने वाले Statements

6.1 सीखने के उद्देश्य

Boolean conditions के आधार पर program execution नियंत्रित करना सीखेंगे। if, if-else, else-if ladder, nested if और traditional switch का उपयोग समझेंगे।

6.2 Conditional Statement

Conditional statement condition के अनुसार code चुनता है। Java में if की condition boolean होनी चाहिए। if (1) invalid है। Comparison और logical expressions boolean result देते हैं।

```
int marks = 75;
if (marks >= 40) {
    System.out.println("Passed");
}
```

Condition false हो तो block skip होता है।

6.3 if-else

```
public class ResultDemo {
    public static void main(String[] args) {
        int marks = 35;
        if (marks >= 40) {
            System.out.println("Pass");
        } else {
            System.out.println("Fail");
        }
    }
}
```

Output: Fail

एक evaluation में दो alternatives में से एक block चलता है। Braces कई statements को block में रखती हैं। Braces न होने पर केवल अगला statement नियंत्रित होता है।

6.4 else-if Ladder

```
int marks = 78;
if (marks < 0 || marks > 100) {
    System.out.println("Invalid marks");
} else if (marks >= 75) {
    System.out.println("Distinction");
} else if (marks >= 60) {
```

```
        System.out.println("First Division");
    } else if (marks >= 40) {
        System.out.println("Pass");
    } else {
        System.out.println("Fail");
    }
}
```

Output: Distinction

पहली true condition चुनी जाती है। Higher category की condition बाद में हो और lower condition पहले true हो जाए, तो higher category तक control नहीं पहुँचेगा। Grading limits यहाँ example के लिए हैं।

6.5 Nested if

```
int age = 20;
boolean documentsComplete = true;
if (age >= 18) {
    if (documentsComplete) {
        System.out.println("Eligible");
    } else {
        System.out.println("Documents required");
    }
} else {
    System.out.println("Age requirement not satisfied");
}
```

Nested decision में inner condition केवल outer block execute होने पर जाँची जाती है। केवल combined eligibility चाहिए, तो age >= 18 && documentsComplete पर्याप्त है।

6.6 Independent if और Ladder

Independent if statements सभी conditions जाँच सकती हैं। किसी number का positive और multiple of five दोनों होना संभव है। Ladder में केवल पहला matching block चुना जाता है।

6.7 Traditional switch

```
int day = 2;
switch (day) {
    case 1:
        System.out.println("Monday");
        break;
    case 2:
        System.out.println("Tuesday");
        break;
    case 3:
        System.out.println("Wednesday");
        break;
    default:
        System.out.println("Other day");
}
```

Output: Tuesday

इस traditional form में byte, short, char, int, संबंधित wrappers, String और enum selectors समझे जाते हैं। Long, float, double और boolean इस form के selector types नहीं हैं। Newer language forms के additional features इस basic chapter में शामिल नहीं हैं।

6.8 Case, break और default

Case matching label है। Duplicate labels invalid हैं। Traditional case values उपयुक्त constants या enum constants होती हैं। Default no-match alternative है और optional है। Break switch से बाहर निकालता है।

```
int choice = 1;
switch (choice) {
    case 1:
        System.out.println("One");
    case 2:
        System.out.println("Two");
        break;
    default:
        System.out.println("Other");
}
```

Output: One फिर Two। इसे fall-through कहते हैं।

6.9 Common Mistakes और Revision

गलती	परिणाम
if(condition);	Empty statement; अगला block स्वतंत्र हो जाता है
Ranges का गलत क्रम	गलत category
Missing break	Fall-through संभव
Numeric if condition	Compilation error
Ambiguous unbraced nesting	Else nearest unmatched if से जुड़ेगा

अभ्यास: Three numbers में largest, leap year check और marks classification के programs बनाएँ।
Leap year rule: divisible by 400, या divisible by 4 लेकिन 100 से नहीं।

MCQs: 1. if condition: A. int B. boolean C. String D. char 2. Ladder में: A. सभी blocks B. पहला matching block C. हमेशा दो D. केवल else 3. Missing break से: A. Fall-through B. हमेशा compile error C. Object deletion D. JVM stop

उत्तर: 1-B, 2-B, 3-A.

UNIT I / CHAPTER 07

अध्याय 7: Looping Statements

Repetition, break एवं continue

7.1 Loop की अवधारणा

Loop किसी block को condition या iteration structure के अनुसार दोहराता है। मुख्य भाग initialization, condition, body और update हैं। Repetition से code छोटा और systematic बनता है।

7.2 while Loop

Entry-controlled loop है। पहले condition फिर body चलती है।

```
int i = 1;
while (i <= 5) {
    System.out.println(i);
    i++;
}
```

Output: 1, 2, 3, 4 और 5 अलग lines में। Initial condition false हो तो zero iterations।

7.3 do-while Loop

Exit-controlled loop में body पहले और condition बाद में जाँची जाती है।

```
int number = 10;
do {
    System.out.println(number);
    number++;
} while (number <= 5);
```

Output: 10

कम-से-कम एक iteration होती है। अंत का semicolon आवश्यक है। Menu-driven repetition में यह उपयोगी हो सकता है।

7.4 for Loop

```
for (int i = 1; i <= 5; i++) {
    System.out.println(i);
}
```

Initialization एक बार, फिर condition, body, update का क्रम है। Update के बाद condition फिर जाँची जाती है।

आधार	while	do-while	for
Test	पहले	बाद में	पहले
Minimum iterations	0	1	0
सामान्य उपयोग	Condition-based	At least once	Counter-based

7.5 Sum Program

```
public class SumDemo {
    public static void main(String[] args) {
        int sum = 0;
        for (int i = 1; i <= 5; i++) {
            sum += i;
        }
        System.out.println("Sum: " + sum);
    }
}
```

Output: Sum: 15

First n natural numbers का sum $n(n+1)/2$ है। Loop calculation और formula से result cross-check किया जा सकता है।

7.6 break और continue

```
for (int i = 1; i <= 5; i++) {
    if (i == 3) {
        break;
    }
    System.out.println(i);
}
```

Output: 1 और 2।

Break की जगह continue हो, तो output 1,2,4,5 होगा। Break nearest enclosing loop/switch से बाहर करता है। Continue current iteration का बाकी भाग छोड़ता है। For में continue के बाद update फिर condition होती है। While में update skip होने से infinite loop का खतरा है।

7.7 Nested Loops

```
for (int row = 1; row <= 3; row++) {
    for (int col = 1; col <= 4; col++) {
        System.out.print("* ");
    }
    System.out.println();
}
```

Output: तीन rows, प्रत्येक में चार stars। Inner body कुल $3 \times 4 = 12$ बार चलती है।

7.8 Enhanced for

```
int[] marks = {70, 80, 90};
for (int value : marks) {
    System.out.println(value);
}
```

Arrays और iterable collections को traverse करता है। Primitive array में loop variable बदलने से original element नहीं बदलता। Direct index चाहिए तो ordinary for उपयोग करें।

7.9 Factorial

```
public class FactorialDemo {
    public static void main(String[] args) {
        int number = 5;
        long factorial = 1;
        if (number < 0 || number > 20) {
            System.out.println("Use a value from 0 to 20.");
            return;
        }
        for (int i = 2; i <= number; i++) {
            factorial *= i;
        }
        System.out.println("Factorial: " + factorial);
    }
}
```

Output: Factorial: 120। $0! = 1$; $21!$ signed long में नहीं समाता।

7.10 Infinite Loops और अभ्यास

`while(true)` या `for(;;)` infinite repetition बना सकते हैं। Break या return से बाहर निकल सकते हैं। Bounds में एक की गलती off-by-one error है।

अभ्यास: Table of a number, reverse counting, digit sum, prime checking और triangular star pattern।

MCQs: 1. At least one iteration: A. while B. for C. do-while D. enhanced for 2. Current iteration skip: A. return B. continue C. new D. import 3. Outer 4, inner प्रत्येक बार 3: A. 7 B. 12 C. 3 D. 4

उत्तर: 1-C, 2-B, 3-B.

UNIT I / CHAPTER 08

अध्याय 8: Arrays

One-Dimensional, Multidimensional एवं Jagged Arrays

8.1 Array क्या है?

Array एक object है जो निश्चित संख्या में एक component type के elements रखता है। Creation के बाद length नहीं बदलती, लेकिन elements बदल सकते हैं। Indexing zero से शुरू होती है।

8.2 Declaration और Creation

```
int[] marks;           // Declaration
marks = new int[5];     // Creation
int[] scores = {70, 80, 90}; // Initialization
```

Declaration अकेले array object नहीं बनाती। Int array elements defaults में 0 होते हैं। Reference array elements null होते हैं।

पहले से declared variable को initializer देने का रूप:

```
marks = new int[]{60, 75, 85};
```

8.3 Index और length

```
System.out.println(scores[0]); // 70
scores[1] = 85;
System.out.println(scores.length); // 3
```

Length n हो तो valid indices 0 से n-1 हैं। Negative या excessive index `ArrayIndexOutOfBoundsException` देता है। Zero-length arrays valid हैं।

Exam Point: Array में length field है; String में length() method है।

8.4 Traversal

```
for (int i = 0; i < scores.length; i++) {
    System.out.println(scores[i]);
}
```

Enhanced for से read कर सकते हैं। Original primitive elements modify करने के लिए index-based assignment उपयोग करें।

8.5 Sum, Average एवं Maximum

```
public class ArrayStatistics {
    public static void main(String[] args) {
        int[] marks = {70, 85, 60, 95, 80};
        int sum = 0;
        int maximum = marks[0];
        for (int value : marks) {
            sum += value;
            if (value > maximum) {
                maximum = value;
            }
        }
        double average = (double) sum / marks.length;
        System.out.println("Sum: " + sum);
        System.out.println("Average: " + average);
        System.out.println("Maximum: " + maximum);
    }
}
```

Output: Sum: 390; Average: 78.0; Maximum: 95। यह example nonempty array मानता है।

8.6 Linear Search

```
int[] values = {12, 25, 37, 48};
int target = 37;
int position = -1;
for (int i = 0; i < values.length; i++) {
    if (values[i] == target) {
        position = i;
        break;
    }
}
System.out.println(position); // 2
```

Worst case में n elements check करने पड़ते हैं; time complexity $O(n)$ है। -1 यहाँ no-match indicator है।

8.7 Two-Dimensional Array

Java में 2D array arrays का array है।

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6}
};
for (int row = 0; row < matrix.length; row++) {
    for (int col = 0; col < matrix[row].length; col++) {
        System.out.print(matrix[row][col] + " ");
    }
    System.out.println();
}
```

Output: पहली row 1 2 3; दूसरी row 4 5 6। matrix.length = 2; matrix[0].length = 3।

8.8 Jagged Arrays

```
int[][] values = {  
    {1, 2},  
    {3, 4, 5},  
    {6}  
};
```

Rows की lengths अलग हैं। Inner loop में प्रत्येक row का length उपयोग करें। कोई row null भी हो सकती है यदि initialize नहीं हुई हो; access से पहले आवश्यक validation करें।

8.9 Reference Assignment एवं Copy

```
int[] first = {10, 20};  
int[] second = first;  
second[0] = 99;  
System.out.println(first[0]); // 99
```

दोनों एक array को refer करते हैं। `first.clone()` अलग array बनाती है। Object references के array में clone shallow copy है; referenced objects साझा रह सकते हैं।

8.10 Revision एवं अभ्यास

Array length fixed है। `i <= length` सामान्य traversal में गलत है। Index loop elements बदल सकता है। Array reference assignment independent copy नहीं है।

अभ्यास: Minimum, reverse display, occurrences count, linear search और 2×2 matrix addition।

MCQs: 1. 5 elements का last index: A. 5 B. 4 C. 6 D. 1 2. Int array default element: A. null B. 1 C. 0 D. Undefined 3. 2D array: A. Arrays का array B. Primitive C. String D. हमेशा square

उत्तर: 1-B, 2-C, 3-A.

UNIT I / CHAPTER 09

अध्याय 9: Classes, Objects एवं Methods

Object-Oriented Programming की आधारभूत अवधारणाएँ

9.1 Class और Object

Class objects की state और behavior का declaration है। Fields state रखते हैं और methods behavior देते हैं। Object runtime instance है।

```
class Student {
    String name;
    int marks;
    void display() {
        System.out.println(name + ": " + marks);
    }
}
```

```
Student student = new Student();
student.name = "Amit";
student.marks = 85;
student.display();
```

Output: Amit: 85

Student type है; student reference variable है; new object creation और constructor invocation करता है। Reference तथा object एक चीज नहीं हैं।

9.2 अनेक Objects

```
class Student {
    String name;
    int marks;
    void display() {
        System.out.println(name + ": " + marks);
    }
}

public class StudentDemo {
    public static void main(String[] args) {
        Student first = new Student();
        first.name = "Amit";
        first.marks = 85;
        Student second = new Student();
        second.name = "Neha";
        second.marks = 92;
        first.display();
        second.display();
    }
}
```

Save as StudentDemo.java | **Output:** Amit: 85; Neha: 92 | Instance fields दोनों objects की अलग हैं।

9.3 Methods और Parameters

```
int add(int first, int second) {
    return first + second;
}
```

Return type int है। First और second declaration के parameters हैं। Call `add(10, 20)` में 10 और 20 arguments हैं। Return execution समाप्त करके result caller को देता है। Void method कोई value return नहीं करती, लेकिन empty `return;` उपयोग कर सकती है।

9.4 Static और Instance Methods

आधार	Instance	Static
संबंध	Object	Class
सामान्य call	<code>object.method()</code>	<code>ClassName.method()</code>
this उपलब्ध	हाँ	नहीं
Instance fields	Direct access संभव	Object reference चाहिए

```
public class Calculator {
    static int square(int number) {
        return number * number;
    }
    public static void main(String[] args) {
        System.out.println(Calculator.square(5));
    }
}
```

Output: 25

9.5 Method Overloading

Same method name, अलग parameter lists:

```
static int add(int a, int b) { return a + b; }
static double add(double a, double b) { return a + b; }
static int add(int a, int b, int c) { return a + b + c; }
```

Parameter count, types या type order बदल सकता है। केवल return type बदलना पर्याप्त नहीं है। Compiler applicable overload चुनता है; ambiguous calls compilation error दे सकती हैं।

9.6 Pass-by-Value

Java हमेशा pass-by-value है। Primitive argument की value copy होती है। Reference argument में reference value copy होती है।

```
static void change(int value) {  
    value = 100;  
}  
static void changeArray(int[] values) {  
    values[0] = 100;  
}
```

पहला caller का primitive variable नहीं बदलता। दूसरा उसी array को modify करता है। Parameter को नए array पर reassign करने से caller का reference नहीं बदलता।

9.7 Encapsulation और Access

Encapsulation state और operations को व्यवस्थित करके access नियंत्रित करने में सहायता करती है।

```
class Account {  
    private double balance;  
    public double getBalance() {  
        return balance;  
    }  
}
```

Public accessible API; no modifier package access; protected same package और cross-package subclasses में नियमों के अनुसार; private class nest की internal access देता है। Beginner use में private fields को बाहर से direct access रोकने वाला समझें।

9.8 Revision और अभ्यास

Class blueprint जैसी संरचना है; object instance है। Method और field access में dot उपयोग होता है। Static members class से संबंधित हैं। Java objects के लिए भी pass-by-value है।

अभ्यास: Book, Employee और Rectangle classes बनाएँ। Parameterized methods और overloading दिखाएँ।

MCQs: 1. Object keyword: A. class B. new C. static D. return 2. केवल return type बदलना: A. Valid overloading B. पर्याप्त नहीं C. Constructor D. Package 3. Java argument passing: A. Reference B. Value C. Name D. Pointer-only

उत्तर: 1-B, 2-B, 3-B.

UNIT I / CHAPTER 10

अध्याय 10: Constructors

Object Initialization एवं Constructor Chaining

10.1 Constructor की परिभाषा

Constructor object creation के दौरान invoke होने वाला विशेष declaration है। यह object की initial state सेट करने में सहायता करता है। नाम class जैसा होता है और कोई return type नहीं होता, void भी नहीं।

Constructors overload हो सकते हैं, inherited नहीं होते और static, abstract या final नहीं हो सकते।

10.2 No-Argument Constructor

```
class Student {  
    String name;  
    Student() {  
        name = "Unknown";  
    }  
}
```

No-argument constructor की parameter list empty है। Programmer इसे स्वयं लिख सकता है।

10.3 Default Constructor

यदि class में कोई constructor घोषित नहीं है, तो compiler default constructor देता है।

```
class Student {  
    int marks;  
}
```

लेकिन Student(int marks) घोषित हो तो automatic no-argument constructor नहीं मिलता।

शब्द	अर्थ
Default constructor	Compiler-provided
No-argument constructor	Empty parameters; user-written भी हो सकता है

10.4 Parameterized Constructor और this

```
class Student {  
    String name;  
    int marks;  
    Student(String name, int marks) {  
        this.name = name;  
        this.marks = marks;  
    }  
}
```

```

    }
}

```

This current object का reference है। this.name field और name parameter है। Static context में this नहीं होता।

10.5 Overloading और Chaining

```

class Student {
    String name;
    int marks;
    Student() {
        this("Unknown", 0);
    }
    Student(String name) {
        this(name, 0);
    }
    Student(String name, int marks) {
        this.name = name;
        this.marks = marks;
    }
    void display() {
        System.out.println(name + ": " + marks);
    }
}

public class ConstructorDemo {
    public static void main(String[] args) {
        Student first = new Student();
        Student second = new Student("Neha", 92);
        first.display();
        second.display();
    }
}

```

Output: Unknown: 0; Neha: 92। Save as ConstructorDemo.java।

This(...) उसी class का constructor और super(...) superclass का constructor invoke करता है। इस पुस्तक में explicit invocation पहले statement में रखी गई है, जो traditional versions के अनुरूप है। Recursive constructor chaining स्वीकार्य नहीं है। Accessible superclass no-argument constructor न हो तो suitable explicit invocation आवश्यक हो सकती है।

10.6 Copy-Style Constructor

Java automatic copy constructor नहीं देती। Programmer लिख सकता है:

```

Student(Student other) {
    this.name = other.name;
    this.marks = other.marks;
}

```

Reference fields की copy shallow हो सकती है। Independent mutable objects चाहिए तो अलग copying logic चाहिए।

10.7 Constructor बनाम Method

आधार	Constructor	Method
नाम	Class जैसा	Valid method name
Return type	नहीं	Type या void
उद्देश्य	Initialization	Behavior
Invocation	Creation/chaining	Method call
Inheritance	नहीं	नियमों के अनुसार

`void Student(){} एक method है, constructor नहीं। Constructors fields को automatic default values मिलने के बाद initialization logic कर सकते हैं।`

10.8 Revision एवं अभ्यास

Default और no-argument में अंतर रखें। This current object; this(...) constructor invocation है। Superclass initialization object construction का भाग है।

अभ्यास: Rectangle के overloaded constructors, Book का parameterized constructor और copy-style constructor बनाएँ।

MCQs: 1. Constructor return type: A. void B. int C. कोई नहीं D. Class 2. Default constructor कब: A. हमेशा B. No declared constructor C. केवल public D. दो constructors 3. this(...): A. Same-class constructor B. JVM C. Package D. हमेशा superclass

उत्तर: 1-C, 2-B, 3-A.

UNIT I / CHAPTER 11

अध्याय 11: Nested एवं Inner Classes

Class के भीतर Class

11.1 Nested Classes

Class के भीतर घोषित class nested class है। Related implementation और helper types को enclosing class के साथ रखने में उपयोगी है। Static nested class inner class नहीं है।

प्रकार	स्थान
Static nested	Class member, static
Member inner	Class member, non-static
Local	Method या block
Anonymous	Expression में unnamed implementation

11.2 Static Nested Class

```
class Outer {
    static int value = 10;
    static class Nested {
        void display() {
            System.out.println(value);
        }
    }
}

public class StaticNestedDemo {
    public static void main(String[] args) {
        Outer.Nested object = new Outer.Nested();
        object.display();
    }
}
```

Output: 10

Outer instance की आवश्यकता नहीं है। Outer instance field access करने के लिए object reference चाहिए।

11.3 Member Inner Class

```
class Outer {
    private int value = 25;
    class Inner {
        void display() {
            System.out.println(value);
        }
    }
}
```

```
    }  
}  
public class InnerDemo {  
    public static void main(String[] args) {  
        Outer outer = new Outer();  
        Outer.Inner inner = outer.new Inner();  
        inner.display();  
    }  
}
```

Output: 25

Inner object enclosing instance से जुड़ा है। Enclosing private members access कर सकता है।
Outer.this outer instance स्पष्ट करता है।

11.4 Local Class

```
public class LocalClassDemo {  
    public static void main(String[] args) {  
        int marks = 85;  
        class Result {  
            void display() {  
                System.out.println("Marks: " + marks);  
            }  
        }  
        new Result().display();  
    }  
}
```

Output: Marks: 85

Captured local variable final या effectively final होनी चाहिए। Effectively final का अर्थ initialization के बाद reassign नहीं हुई, भले final keyword न लिखा हो।

11.5 Anonymous Class

```
interface Greeting {  
    void sayHello();  
}  
public class AnonymousDemo {  
    public static void main(String[] args) {  
        Greeting greeting = new Greeting() {  
            @Override  
            public void sayHello() {  
                System.out.println("Namaste!");  
            }  
        };  
        greeting.sayHello();  
    }  
}
```

Output: Namaste!

यह interface implement करने वाली unnamed class का object है, interface का direct object नहीं। Explicit named constructor नहीं लिख सकते। Captured locals के लिए final/effectively-final rule लागू है।

11.6 उपयोग एवं Comparison

Static nested helper को outer object नहीं चाहिए। Member inner को enclosing instance चाहिए। Local class का नाम declaration scope में उपलब्ध है। Anonymous class one-time implementation में उपयोगी है। बहुत nesting readability घटा सकती है।

विषय	Static Nested	Member Inner
Outer object	आवश्यक नहीं	सामान्य creation में आवश्यक
Creation	<code>new Outer.Nested()</code>	<code>outer.new Inner()</code>
Outer instance access	Reference चाहिए	Associated object उपलब्ध

Inner classes के static declaration rules Java versions में बदले हैं; यहाँ basic compatible patterns उपयोग किए गए हैं।

अभ्यास: College.Department inner class तथा anonymous greeting बनाएँ। Types का comparison लिखें।

MCQs: 1. Inner नहीं है: A. Member inner B. Static nested C. Local inner D. Anonymous inner 2. Creation: A. `outer.new Inner()` B. `new inner.Outer()` C. `Inner.static()` D. `package.Inner()` 3. Captured local: A. हमेशा public B. final/effectively final C. हमेशा array D. हमेशा static

उत्तर: 1-B, 2-A, 3-B.

UNIT I / CHAPTER 12

अध्याय 12: String Handling

Text Processing, StringBuilder एवं StringBuffer

12.1 String

String Java की immutable class है जो text sequence represent करती है। `String text="Java";` में literal से String reference मिलता है। Concatenation result assign करने से reference बदल सकता है; original String का content नहीं बदलता।

12.2 Literal और Constructor

```
String a = "Java";  
String b = "Java";  
String c = new String("Java");  
System.out.println(a == b);    // true  
System.out.println(a == c);    // false  
System.out.println(a.equals(c)); // true
```

Literals interned होते हैं। `==` reference identity और `equals` content जाँचता है। String equality के लिए `==` पर निर्भर न रहें।

12.3 प्रमुख Methods

मान लें `String text = "Java Programming";`

Expression	Result
<code>text.length()</code>	16
<code>text.charAt(0)</code>	J
<code>text.substring(0, 4)</code>	Java
<code>text.substring(5)</code>	Programming
<code>text.indexOf("Programming")</code>	5
<code>text.contains("Java")</code>	true
<code>text.startsWith("Java")</code>	true
<code>text.endsWith("ing")</code>	true
<code>text.replace("Java", "Python")</code>	Python Programming

Substring में begin included और end excluded है। Invalid indices exceptions दे सकते हैं। String length UTF-16 code units की संख्या है, हमेशा visible characters की संख्या नहीं।

12.4 Comparison एवं Case

```
System.out.println("JAVA".equalsIgnoreCase("java"));
System.out.println("Apple".compareTo("Banana"));
System.out.println("Java".compareTo("Java"));
```

Output: true, -1, 0। CompareTo lexicographical negative/zero/positive result देता है, केवल -1/0/1 आवश्यक नहीं। Case conversion locale-dependent हो सकती है।

12.5 Spaces, split और null

Trim शुरुआत/अंत के U+0020 तक के characters हटाता है, हर Unicode whitespace नहीं। Split regular expression उपयोग करता है।

```
String[] subjects = "Java,Python,C".split(",");
for (String subject : subjects) {
    System.out.println(subject);
}
```

Output: Java, Python, C अलग lines में।

Empty String "" एक object है जिसकी length 0 है। Null का अर्थ object reference नहीं। Null पर instance method call NullPointerException देता है। "Java".equals(value) null-safe comparison है।

12.6 StringBuilder

Mutable character sequence है; repeated text building में उपयोगी है।

```
StringBuilder builder = new StringBuilder("Java");
builder.append(" Programming");
builder.insert(0, "Learn ");
System.out.println(builder);
```

Output: Learn Java Programming

Method	कार्य
append	अंत में जोड़ना
insert	Position पर जोड़ना
delete	Range हटाना
replace	Range बदलना

Method	कार्य
reverse	Sequence reverse
toString	String result

12.7 StringBuffer

Mutable है और प्रमुख methods synchronized हैं। StringBuilder synchronization नहीं देता। Synchronized individual calls का अर्थ पूरे multi-call operation का atomic होना नहीं है।

Class	Mutable?	सामान्य उपयोग
String	नहीं	स्थिर text
StringBuilder	हाँ	Single-thread text building
StringBuffer	हाँ	Synchronized mutable text operations

12.8 Palindrome Program

```
public class PalindromeDemo {
    public static void main(String[] args) {
        String text = "madam";
        String reversed = new StringBuilder(text)
            .reverse().toString();
        if (text.equals(reversed)) {
            System.out.println("Palindrome");
        } else {
            System.out.println("Not a palindrome");
        }
    }
}
```

Output: Palindrome। यह exact comparison है; spaces और case automatically ignore नहीं होते।

12.9 Common Mistakes एवं अभ्यास

Concat result बिना assignment original variable नहीं बदलता। Array length और String length() अलग हैं। Null और empty text अलग हैं। StringBuilder को String content के समान equals behavior वाला न मानें; content compare करने के लिए toString results उपयोग कर सकते हैं।

अभ्यास: Vowel count, reverse, palindrome और word splitting।

MCQs: 1. Content equality: A. = B. equals C. केवल == D. % 2. "Java".substring(1,3): A. av B. ava C. Ja D. va 3. Mutable synchronized methods: A. String B. StringBuffer C. Integer D. Character

उत्तर: 1-B, 2-A, 3-B.

UNIT I / CHAPTER 13

अध्याय 13: Packages

Built-In एवं User-Defined Packages

13.1 Package की अवधारणा

Package related types को namespace में व्यवस्थित करता है। Organization, same simple names के separation और access control में उपयोगी है। Package केवल folder नहीं है, लेकिन matching folder structure सामान्य project convention है।

13.2 Built-In Packages

Package	प्रमुख उपयोग
java.lang	String, System, Math
java.util	Scanner, Arrays, ArrayList
java.io	Input/output
java.nio.file	Files और paths
java.time	Date/time
java.net	Networking

Java.lang automatically imported है। Subpackages automatically import नहीं होतीं।

13.3 Imports

```
import java.util.Scanner;
```

Accessible type को simple name से उपयोग कर सकते हैं। Import class loading command नहीं है। Fully qualified name भी उपयोग कर सकते हैं।

```
java.util.Scanner input =  
    new java.util.Scanner(System.in);
```

Wildcard `import java.util.*`; उस package की accessible types के names उपलब्ध करता है, subpackages नहीं। `import java.*`; से Scanner import नहीं होती।

13.4 Static Import

```
import static java.lang.Math.sqrt;
public class StaticImportDemo {
    public static void main(String[] args) {
        System.out.println(sqrt(81));
    }
}
```

Output: 9.0

13.5 User-Defined Package

File: src/college/tools/Calculator.java

```
package college.tools;
public class Calculator {
    public static int add(int a, int b) {
        return a + b;
    }
}
```

File: src/college/app/App.java

```
package college.app;
import college.tools.Calculator;
public class App {
    public static void main(String[] args) {
        System.out.println(Calculator.add(10, 20));
    }
}
```

Package declaration comments के बाद और imports/type declarations से पहले आती है।

13.6 Compilation एवं Execution

Project root से:

```
javac -d out src/college/tools/Calculator.java src/college/app/App.java
java -cp out college.app.App
```

Output: 30

Long compile command को terminal में एक line में लिखें। Out classpath root है। Fully qualified name college.app.App है। Classpath को out/college/app बनाने से इस class की lookup गलत होगी।

13.7 Access Control

Public top-level class दूसरे packages में accessible हो सकती है। No-modifier top-level class package-private है। Public class के members भी आवश्यक accessibility रखने चाहिए।

College और college.tools अलग packages हैं। Parent-like naming automatic access privilege नहीं देती। Subpackage parent package का member नहीं है।

13.8 Naming और Common Errors

Package names lowercase convention में: college.tools, college.records। Organizations reverse-domain style उपयोग करती हैं: com.example.application।

समस्या	समाधान
Packaged main class नहीं मिली	FQCN और classpath root जाँचें
Type inaccessible	Class/member access modifier जाँचें
Same simple name conflict	Fully qualified name उपयोग करें
Wildcard से subpackage missing	Required type explicitly import करें

13.9 Quick Revision एवं अभ्यास

Package namespace देता है। Imports compile-time naming सुविधा हैं। java.lang automatic है। -d compiled output बताता है। Same package और subpackage access एक नहीं है।

अभ्यास: College.geometry में Rectangle class बनाएँ। दूसरे package से area call करें। Sources अलग files में रखें और compile commands लिखें।

MCQs: 1. Automatic package: A. java.util B. java.io C. java.lang D. java.net 2. Wildcard subpackages import करता है: A. हाँ B. नहीं C. हमेशा एक D. OS पर निर्भर 3. Packaged launcher name: A. college.app.App B. App.class C. college/app/App.class D. package App

उत्तर: 1-C, 2-B, 3-A.

UNIT I / CHAPTER 14

अध्याय 14: Revision एवं Question Bank

Unit I का समेकित अभ्यास

14.1 Unit Revision Map

Topic	मुख्य बिंदु
Java history	Oak; James Gosling; सार्वजनिक प्रस्तुति 1995
Platform independence	Compatible JVM पर bytecode
JVM / JRE / JDK	Execution / runtime / development
Files	.java source, .class compiled, .jar archive
Types	8 primitives और reference types
Variables	Local, instance, static
Operators	Arithmetic, logical, bitwise, conditional
Selection	if, ladder, switch
Loops	while, do-while, for, enhanced for
Arrays	Fixed length, zero-based indices
Classes	State और behavior
Constructors	Initialization; no return type
Nested classes	Static nested, member inner, local, anonymous
Strings	Immutable; equals for content
Packages	Namespace और organization

14.2 Frequently Confused Concepts

पहला	दूसरा	मुख्य अंतर
=	==	Assignment / equality
&&	&	Short-circuit / non-short-circuit boolean AND

पहला	दूसरा	मुख्य अंतर
break	continue	Exit / current iteration skip
length	length()	Array field / String method
Default constructor	No-argument constructor	Compiler-provided / empty parameters
== references	String equals	Identity / content
Casting	Parsing	Type conversion / numerical text पढ़ना

14.3 Solved Output Questions

Q1: `double x = 9 / 2;` के बाद `print x`? **उत्तर: 4.0**। Integer division पहले हुई।

Q2: `int x=3; int y=x++ + ++x;` के बाद `x` और `y`? **उत्तर: x=5, y=8**।

Q3: `2 + 3 + "4"` और `"2" + 3 + 4`? **उत्तर: 54 और 234**।

Q4: `int x=1; boolean r=true || ++x>1;` में `x`? **उत्तर: 1**। Short-circuit।

Q5: `String s="Java"; s.concat(" Book");` में `s`? **उत्तर: Java**। Returned result assign नहीं किया गया।

Q6: `int[] a={5,10}; int[] b=a; b[1]=99;` में `a[1]`? **उत्तर: 99**। Shared array।

Q7: Outer loop 3, inner प्रत्येक बार 2 iterations? **उत्तर: 6 inner executions**।

Q8: `int x=10; x+=2.9;` में `x`? **उत्तर: 12**। Compound assignment conversion।

14.4 MCQ Question Bank

1. Java का प्रारंभिक नाम? A. Oak B. Pine C. BASIC D. Pascal
2. Source compiler? A. java B. javac C. jar D. jdb
3. Reference type? A. int B. char C. String D. boolean
4. Long size? A. 8 B. 16 C. 32 D. 64 bits
5. Valid declaration? A. `float x=2.5F` B. `char x="A"` C. `boolean x=1` D. `byte x=200`
6. 010 की decimal value? A. 10 B. 8 C. 2 D. 16
7. `(int)8.9`? A. 9 B. 8 C. 8.9 D. Error
8. Remainder operator? A. / B. % C. ^ D. ~
9. ^ operation? A. Power B. XOR C. Division D. Assignment
10. Short-circuit operator? A. & B. | C. && D. ^
11. At least one iteration? A. while B. for C. do-while D. enhanced for
12. Six elements का last index? A. 6 B. 5 C. 7 D. 0
13. Array size? A. length B. length() C. size() D. count
14. Object creation? A. static B. new C. return D. import
15. Constructor name समान है? A. Package B. Class C. Method D. Extension

16. Default constructor कब? A. हमेशा B. कोई declared constructor नहीं C. केवल parameterized होने पर D. कभी नहीं
17. Static context में current object keyword उपलब्ध नहीं? A. this B. new C. class D. return
18. Argument passing? A. Reference B. Value C. Name D. Pointer only
19. String content comparison? A. = B. equals C. केवल == D. %
20. Immutable class? A. String B. StringBuilder C. StringBuffer D. हर array
21. "Java".length()? A. 3 B. 4 C. 5 D. 0
22. "Java".charAt(1)? A. J B. a C. v D. A
23. Automatic import? A. java.util B. java.io C. java.lang D. java.net
24. JAR full form? A. Java Archive B. Java Array Runtime C. Joint Resource D. Java Record
25. null instanceof String? A. true B. false C. Exception D. 1

14.5 Answer Key

प्रश्न	उत्तर	प्रश्न	उत्तर	प्रश्न	उत्तर
1	A	10	C	19	B
2	B	11	C	20	A
3	C	12	B	21	B
4	D	13	A	22	B
5	A	14	B	23	C
6	B	15	B	24	A
7	B	16	B	25	B
8	B	17	A	-	-
9	B	18	B	-	-

14.6 Short-Answer Questions

1. WORA और compatible JVM समझाइए।
2. JVM, JRE और JDK में अंतर लिखिए।
3. Bytecode क्या है?
4. Primitive और reference types में अंतर बताइए।
5. Local variable initialization क्यों आवश्यक है?
6. Final reference और immutable object का अंतर लिखिए।
7. Short-circuit evaluation क्या है?
8. Break और continue का अंतर बताइए।

9. Constructor और method का अंतर लिखिए।
10. Packages का उद्देश्य बताइए।

14.7 Long-Answer Questions

1. Java features और execution process उदाहरण सहित समझाइए।
2. Data types, literals और conversions का वर्णन कीजिए।
3. Operators numerical examples सहित समझाइए।
4. Conditional और looping statements की तुलना कीजिए।
5. One-dimensional और jagged arrays समझाइए।
6. Class, object, method और constructor का संबंध बताइए।
7. Nested classes के प्रकार examples सहित समझाइए।
8. String methods और mutable text classes की तुलना कीजिए।
9. User-defined package compile/run करने के चरण लिखिए।

14.8 Mini Project: Student Result Application

प्रत्येक subject maximum 100 और passing marks 40 मानें। Nonempty array तथा marks की range validate करें। Defensive copy caller के array से अलग state रखती है।

```
class Student {
    private final String name;
    private final int[] marks;

    Student(String name, int[] marks) {
        if (marks == null || marks.length == 0) {
            throw new IllegalArgumentException("Marks required");
        }
        for (int mark : marks) {
            if (mark < 0 || mark > 100) {
                throw new IllegalArgumentException("Use 0-100 marks");
            }
        }
        this.name = name;
        this.marks = marks.clone();
    }

    int total() {
        int sum = 0;
        for (int mark : marks) {
            sum += mark;
        }
        return sum;
    }

    double percentage() {
        return (double) total() / marks.length;
    }

    boolean passed() {
        for (int mark : marks) {
            if (mark < 40) {
                return false;
            }
        }
        return true;
    }
}
```

```

        }
    }
    return true;
}

void display() {
    System.out.println("Name: " + name);
    System.out.println("Total: " + total());
    System.out.println("Percentage: " + percentage());
    System.out.println("Result: " + (passed() ? "Pass" : "Fail"));
}
}

public class StudentResultApp {
    public static void main(String[] args) {
        int[] marks = {75, 80, 65, 90, 70};
        Student student = new Student("Amit", marks);
        student.display();
    }
}

```

14.9 Compile और Output

```

javac StudentResultApp.java
java StudentResultApp

```

```

Name: Amit
Total: 380
Percentage: 76.0
Result: Pass

```

Total = 380; maximum = 500; percentage = $380/500 \times 100 = 76$ । All subjects 40 या अधिक हैं, इसलिए pass। Exception handling का विस्तार आगे की units में पढ़ा जा सकता है।

14.10 Project Extensions

- Scanner से name और marks प्राप्त करें।
- अनेक students की list बनाएँ।
- Highest/lowest subject marks निकालें।
- Classes अलग packages में रखें।
- Executable JAR तैयार करें।

14.11 अंतिम Revision Checklist

क्या आप बिना सहायता Hello program compile कर सकते हैं? क्या integer division, prefix/postfix और array indexing का output बता सकते हैं? क्या class/constructor का अंतर स्पष्ट है? क्या String comparison और package launch commands सही लिख सकते हैं? जहाँ उत्तर अस्पष्ट हो, संबंधित chapter का program स्वयं चलाकर परिणाम जाँचें।