freelance_Project available to buy contact on 8007592194

SR.NO	Project NAME	Technology
1	E-Learning HUB	React+Springboot+MySql
2	PG MATES	React+Springboot+MySql
3	Tour and Travel	React+Springboot+MySql
4	Marriage Hall booking	React+Springboot+MySql
5	Bus ticket booking Mini Project	React+Springboot+MySql
6	Quizz App /Exam Portal Mini Project	Springboot,MySql,JSP,Html
7	Event Management System	React+Springboot+MySql
8	Hotel Mangement System	React+Springboot+MySql
9	Agriculture Web Project	React+Springboot+MySql
10	AirLine Reservation System	React+Springboot+MySql
11	E-Commerce Web Project	React+Springboot+MySql
12	Sport Ground Booking	React+Springboot+MySql
13	CharityDonation web project	React+Springboot+MySql
14	Hospital Management Project	React+Springboot+MySql
15	Online voting System Mini project	Springboot,MySql,JSP,Html
16	E-Commerce shop mini project	Springboot,MySql,JSP,Html
17	Job Portal web project	React+Springboot+MySql
18	Insurance policy Portal	React+Springboot+MySql
19	Transpotation Services portal	React+Springboot+MySql
20	E-RTO Driving licence portal	React+Springboot+MySql
21	doctor Appointment Portal	React+Springboot+MySql
22	Online food delivery Project	React+Springboot+MySql
23	Muncipal Corporation Management	React+Springboot+MySql
24	E-College Portal Project	React+Springboot+MySql
25	Gym Management	React+Springboot+MySql
X 26	Bike Booking System Portal	React+Springboot+MySql
27	Food Waste Management Portal	React+Springboot+MySql
28	Online Pizza delivery Portal	React+Springboot+MySql
29	Fruite Delivery portal	React+Springboot+MySql
30	HomeRental Booking Project	React+Springboot+MySql
31	FarmerMarketplace	React+Springboot+MySql

OOPs WITH JAVA MCQ BANK
Total No. Of MCQs : 754
Name : Shubham Shivaji Patil

1. Lecture: Introduction to Java

1.1 Features of Java:

1. Which of the following is not a feature of Java?
   a) Platform-independent
   b) Object-oriented
   c) Compiled language
   d) Automatic garbage collection
   Answer: c) Compiled language

2. Java is considered platform-independent because:
   a) It can run only on Windows platforms
   b) It can run on any platform with a Java Virtual Machine (JVM)
   c) It can run on any platform without a JVM
   d) It can run on any platform with .NET framework installed
   Answer: b) It can run on any platform with a Java Virtual Machine (JVM)

3. Which feature in Java ensures that a class cannot be inherited?
   a) Encapsulation
   b) Polymorphism
   c) Abstraction
   d) Final keyword
   Answer: d) Final keyword

4. Java supports multiple inheritance through:
   a) Interfaces
   b) Abstract classes
   c) Enums
   d) Inner classes
   Answer: a) Interfaces

5. The feature in Java that allows a subclass to provide a specific implementation of a method that is already defined in its superclass is called:
   a) Abstraction
   b) Polymorphism
   c) Encapsulation
   d) Inheritance
   Answer: b) Polymorphism

6. Which feature in Java allows you to hide the internal details of a class and show only the functionalities to the outside world?
   a) Abstraction
   b) Encapsulation
   c) Inheritance
   d) Polymorphism
   Answer: b) Encapsulation

7. Which of the following is not a primitive data type in Java?
   a) int
   b) float
   c) string
   d) char

Answer: c) string

8. The primitive data type in Java that can store whole numbers from -128 to 127 is:
a) byte
b) short
c) int
d) long
Answer: a) byte

9. Which primitive data type is used to store floating-point numbers with single-precision?
a) float
b) double
c) decimal
d) real
Answer: a) float

10. What is the default value of the boolean data type in Java?
a) true
b) false
c) null
d) 0
Answer: b) false

1.2 JVM Architecture:
1. JVM stands for:
a) Java Virtual Model
b) Java Virtual Machine
c) Java Visual Machine
d) Java Virtual Memory
Answer: b) Java Virtual Machine

2. Which component of JVM is responsible for converting bytecode to machine-specific code?
a) Class Loader
b) Bytecode Verifier
c) Just-In-Time (JIT) Compiler
d) Garbage Collector
Answer: c) Just-In-Time (JIT) Compiler

3. The JVM memory area that stores method-specific data and local variables is called:
a) Heap
b) Stack
c) Program Counter Register
d) Native Method Stack
Answer: b) Stack

4. Which component of JVM is responsible for loading the required class files during runtime?
a) Class Loader
b) Bytecode Verifier
c) Just-In-Time (JIT) Compiler
d) Garbage Collector
Answer: a) Class Loader

5. The JVM memory area that stores dynamically allocated objects and arrays is called:
a) Heap
b) Stack
c) Program Counter Register
d) Native Method Stack
Answer: a) Heap

6. When does the JVM's garbage collector free up memory?
a) At the end of the program execution
b) After every method call
c) When the program encounters an error
d) When objects are no longer referenced or needed
Answer: d) When objects are no longer referenced or needed

7. The JVM instruction that holds the memory address of the currently-executing JVM instruction is called:
a) Heap
b) Stack
c) Program Counter Register
d) Native Method Stack
Answer: c) Program Counter Register

8. Which component of JVM ensures that the bytecode does not violate Java's security rules?
a) Class Loader
b) Bytecode Verifier
c) Just-In-Time (JIT) Compiler
d) Garbage Collector
Answer: b) Bytecode Verifier

9. The JVM memory area that stores information about native methods is called:
a) Heap
b) Stack
c) Program Counter Register
d) Native Method Stack
Answer: d) Native Method Stack

10. Which of the following is true about JVM?
a) It is platform-dependent
b) It directly executes Java source code
c) It is a physical machine
d) It interprets Java bytecode line-by-line
Answer: d) It interprets Java bytecode line-by-line

1.3 JDK and its Usage:
1. JDK stands for:
a) Java Development Kit
b) Java Deployment Kit
c) Java Debugging Kit
d) Java Documentation Kit
Answer: a) Java Development Kit

2. Which of the following components are included in the JDK?
a) Java Runtime Environment (JRE)
b) Java Compiler (javac)
c) Java Virtual Machine (JVM)
d) JavaFX
Answer: b) Java Compiler (javac)

3. What is the purpose of the Java Runtime Environment (JRE) in the JDK?
a) To compile Java source code
b) To run Java applications and applets
c) To debug Java programs
d) To create Java documentation
Answer: b) To run Java applications and applets

4. Which tool is used to create JAR files in the JDK?
a) jar

b) javac
c) jdb
d) java
Answer: a) jar

5. What is the role of the "java" command in the JDK?
a) To compile Java source code
b) To run Java applications
c) To create JAR files
d) To debug Java programs
Answer: b) To run Java applications

6. Which of the following is not included in the JDK package?
a) JRE (Java Runtime Environment)
b) javac (Java Compiler)
c) JVM (Java Virtual Machine)
d) JAR (Java Archive)
Answer: c) JVM (Java Virtual Machine)

7. The JDK provides tools for developers to perform which tasks?
a) Writing and compiling Java code
b) Running Java applications
c) Debugging Java programs
d) All of the above
Answer: d) All of the above

8. Which JDK tool is used to launch the Java Virtual Machine (JVM)?
a) javac
b) java
c) jar
d) jdb
Answer: b) java

9. Which JDK tool is used for Java program debugging?
a) javac
b) java
c) jar
d) jdb
Answer: d) jdb

10. Which of the following is not a part of the JDK package?
a) JRE (Java Runtime Environment)
b) JavaFX
c) JVM (Java Virtual Machine)
d) JAR (Java Archive)
Answer: b) JavaFX

1.4 Structure of Java Class:
1. In Java, a class is a blueprint for creating:
a) Objects
b) Interfaces
c) Methods
d) Constructors
Answer: a) Objects

2. Which keyword is used to define a class in Java?
a) class
b) interface
c) object
d) new

Answer: a) class

3. Which section of a Java class contains the class variables and constants?
a) Method
b) Constructor
c) Static Block
d) Instance Variables
Answer: d) Instance Variables

4. In Java, a constructor is a special method that is called:
a) When an object is created using the new keyword
b) When a method is executed
c) When the class is loaded by the JVM
d) When an object is destroyed
Answer: a) When an object is created using the new keyword

5. Which of the following is true regarding the main method in a Java class?
a) It is mandatory in every Java class
b) It can have any name other than "main"
c) It must be defined with the "public" access modifier
d) It is called automatically when the program is executed
Answer: a) It is mandatory in every Java class

6. What is the purpose of the static block in a Java class?
a) To create static variables
b) To define static methods
c) To initialize static variables or perform static tasks
d) To create new instances of the class
Answer: c) To initialize static variables or perform static tasks

7. Which keyword is used to create an instance of a class in Java?
a) class
b) object
c) new
d) instance
Answer: c) new

8. The access specifier used to make a method or variable accessible only within the same package is:
a) private
b) public
c) protected
d) package-private (default)
Answer: d) package-private (default)

9. Which section of a Java class contains the code that is executed when an object of the class is created?
a) Method
b) Constructor
c) Static Block
d) Instance Variables
Answer: b) Constructor

10. Can a Java class have multiple constructors?
a) Yes, but only if they have the same number of parameters
b) No, a class can have only one constructor
c) Yes, constructors are not restricted in number or parameters
d) Yes, but only if they have different return types
Answer: c) Yes, constructors are not restricted in number or parameters

1.5 Working with Data Types: Primitive Data Types:
1. Which of the following is a primitive data type in Java?

a) String
b) Integer
c) Boolean
d) Array
Answer: c) Boolean

2. What is the size of the int data type in Java?
a) 4 bytes
b) 8 bytes
c) 2 bytes
d) It varies depending on the JVM
Answer: a) 4 bytes

3. Which primitive data type is used to store numbers with decimal points in Java?
a) int
b) float
c) double
d) decimal
Answer: c) double

4. Which of the following data types is used to store characters in Java?
a) char
b) string
c) character
d) varchar
Answer: a) char

5. What is the default value of the int data type in Java?
a) 0
b) 0.0
c) ""
d) null
Answer: a) 0

6. The boolean data type in Java can have two possible values, which are:
a) True and False
b) Yes and No
c) 0 and 1
d) On and Off
Answer: a) True and False

7. Which data type is used to store true/false values in Java?
a) bool
b) boolean
c) bit
d) tf
Answer: b) boolean

8. The smallest possible value that can be stored in a byte data type in Java is:
a) -128
b) 0
c) 1
d) 127
Answer: a) -128

9. What is the size of the double data type in Java?
a) 4 bytes
b) 8 bytes
c) 2 bytes
d) It varies depending on the JVM

Answer: b) 8 bytes

10. Which data type is used to store whole numbers in Java?
a) int
b) float
c) double
d) char
Answer: a) int

2. Lecture: Operators

2.1 Unary, Binary, Arithmetic, Assignment, Compound, Relational, Logical, Equality
1. Which type of operator requires a single operand and is used for incrementing or decrementing a value?
A) Unary
B) Binary
C) Arithmetic
D) Assignment
Answer: A) Unary

2. The expression `x = y + 5` contains which type of operator?
A) Arithmetic
B) Assignment
C) Relational
D) Logical
Answer: B) Assignment

3. What will be the result of the expression `5 + 3 * 2`?
A) 16
B) 11
C) 13
D) 15
Answer: C) 13

4. The compound assignment operator `+=` can be used to _____.
A) Compare two values for equality
B) Add and assign a value to a variable
C) Multiply two values together
D) Perform logical negation
Answer: B) Add and assign a value to a variable

5. The relational operator `<=` checks whether the left operand is _____ the right operand.
A) Greater than or equal to
B) Less than or equal to
C) Equal to
D) Not equal to
Answer: B) Less than or equal to

2.2 Control Statements

2.2.1 if-else-if:
1. The `if` statement is used to _____.
A) Execute a block of code repeatedly
B) Make a decision based on a condition
C) Declare variables and methods
D) Perform arithmetic operations
Answer: B) Make a decision based on a condition

2. In the context of the `if-else-if` statement, which condition is evaluated if all preceding conditions are false?
   A) First `if` condition
   B) Last `if` condition
   C) First `else` condition
   D) First `else if` condition
   Answer: D) First `else if` condition

3. What will be the output of the following code snippet?
   ```java
   int x = 10;
   if (x > 5) {
      System.out.println("x is greater than 5");
   } else if (x > 0) {
      System.out.println("x is positive");
   } else {
      System.out.println("x is negative");
   }
   ```
   A) x is greater than 5
   B) x is positive
   C) x is negative
   D) The code will not compile
   Answer: A) x is greater than 5

4. The `if-else-if` statement can be used to handle _____.
   A) Multiple conditions, one after the other
   B) Only two mutually exclusive conditions
   C) Recursive function calls
   D) Switch-case scenarios
   Answer: A) Multiple conditions, one after the other

5. What will be the output of the following code snippet?
   ```java
   int age = 25;
   if (age < 18) {
      System.out.println("You are a minor.");
   } else if (age >= 18 && age < 60) {
      System.out.println("You are an adult.");
   } else {
      System.out.println("You are a senior citizen.");
   }
   ```
   A) You are a minor.
   B) You are an adult.
   C) You are a senior citizen.
   D) The code will not compile
   Answer: B) You are an adult.

2.2.2 switch:
1. The `switch` statement in Java is primarily used to _____.
   A) Perform arithmetic operations
   B) Execute a block of code repeatedly
   C) Make decisions based on multiple conditions
   D) Declare variables and methods
   Answer: C) Make decisions based on multiple conditions

2. What is the purpose of the `break` statement inside a `switch` case block?
   A) To terminate the program

B) To exit the loop
C) To move to the next case without executing further code
D) To clear the variable values
Answer: C) To move to the next case without executing further code

3. Which data types can be used as the argument of a `switch` statement?
A) int and String
B) double and char
C) boolean and long
D) float and byte
Answer: A) int and String

4. What will be the output of the following code snippet?
```java
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("Invalid day");
        break;
}
```
A) Monday
B) Tuesday
C) Wednesday
D) Invalid day
Answer: B) Tuesday

5. In a `switch` statement, what happens if there is no `break` statement after a case block?
A) The program will throw a compilation error.
B) The program will enter an infinite loop.
C) The program will execute the next case block as well.
D) The program will terminate abruptly.
Answer: C) The program will execute the next case block as well.

2.2.3 Ternary Operator:
1. The ternary operator in Java has the following syntax:
A) `condition ? expression1 : expression2`
B) `condition ? expression1, expression2`
C) `condition ? expression1 : : expression2`
D) `condition ? : expression1, expression2`
Answer: A) `condition ? expression1 : expression2`

2. The ternary operator is a shorthand for which control statement?
A) if-else-if statement
B) switch statement
C) for loop
D) while loop
Answer: A) if-else-if statement

3. What is the result of the following ternary expression?
`int x = 10;

int y = x > 5 ? x + 2 : x - 2;`
A) x is 12
B) x is 8
C) y is 12
D) y is 8
Answer: C) y is 12

4. The ternary operator is best used when dealing with _____.
A) A single condition and two possible outcomes
B) A single condition and three possible outcomes
C) Multiple conditions and two possible outcomes
D) Multiple conditions and three possible outcomes
Answer: A) A single condition and two possible outcomes

5. What will be the value of `result` after the following code is executed?
```java
int a =

5;
int b = 8;
int result = (a > b) ? a : b;
```
A) 5
B) 8
C) true
D) false
Answer: B) 8

2.2.4 For Loop:
1. The initialization, condition, and iteration expression in a `for` loop are enclosed in _____.
A) Curly braces {}
B) Parentheses ()
C) Square brackets []
D) Angle brackets <>
Answer: B) Parentheses ()

2. How many times will the following `for` loop be executed?
```java
for (int i = 0; i < 5; i++) {
    System.out.println("Hello");
}
```
A) 5 times
B) 6 times
C) 4 times
D) Infinite times (loop never terminates)
Answer: A) 5 times

3. What is the output of the following code snippet?
```java
int sum = 0;
for (int i = 1; i <= 10; i++) {
    sum += i;
}
System.out.println(sum);
```
A) 45
B) 55
C) 100
D) 10

Answer: B) 55

4. The `for` loop can be used to iterate over which of the following data structures?
A) Arrays only
B) Lists only
C) Arrays and Lists
D) Strings only
Answer: C) Arrays and Lists

5. What will be the value of `x` after the execution of the following loop?
```java
int x = 2;
for (int i = 0; i < 5; i++) {
    x *= 2;
}
```
A) 8
B) 16
C) 32
D) 64
Answer: C) 32

2.2.5 While Loop:
1. The `while` loop in Java executes a block of code _____.
A) At least once, regardless of the condition
B) Until the condition becomes false
C) A fixed number of times
D) Indefinitely (loop never terminates)
Answer: B) Until the condition becomes false

2. What is the output of the following code snippet?
```java
int n = 1;
while (n <= 5) {
    System.out.print(n + " ");
    n++;
}
```
A) 1 2 3 4 5
B) 5 4 3 2 1
C) 1 1 1 1 1
D) 5 5 5 5 5
Answer: A) 1 2 3 4 5

3. In the context of a `while` loop, what happens if the initial condition is false?
A) The loop will not be executed.
B) The loop will execute indefinitely.
C) The loop will throw an exception.
D) The loop will execute at least once.
Answer: A) The loop will not be executed.

4. The `while` loop is best used when the number of iterations is _____.
A) Known and fixed
B) Known but variable
C) Unknown and variable
D) Zero
Answer: C) Unknown and variable

5. What will be the value of `sum` after the following loop is executed?
```java
```

```
int sum = 0;
int i = 1;
while (i <= 10) {
    sum += i;
    i++;
}
```
A) 45
B) 55
C) 100
D) 10
Answer: B) 55

2.2.6 Do-While Loop:
1. The `do-while` loop in Java guarantees that the loop body is executed _____.
A) At least once, regardless of the condition
B) Until the condition becomes false
C) A fixed number of times
D) Indefinitely (loop never terminates)
Answer: A) At least once, regardless of the condition

2. What is the output of the following code snippet?
```java
int x = 5;
do {
    System.out.print(x + " ");
    x--;
} while (x > 0);
```
A) 5 4 3 2 1
B) 1 2 3 4 5
C) 5 5 5 5 5
D) The code will not compile
Answer: A) 5 4 3 2 1

3. In a `do-while` loop, the condition is evaluated _____.
A) Before executing the loop body
B) After executing the loop body
C) Twice during each iteration
D) Only once, at the end of the loop
Answer: B) After executing the loop body

4. What will be the value of `product` after the following loop is executed?
```java
int product = 1;
int i = 0;
do {
    product *= i;
    i++;
} while (i <= 5);
```
A) 0
B) 1
C) 120
D) 720
Answer: B) 1

5. The `do-while` loop is useful when you want to ensure that the loop body is executed _____.
A) Only once
B) A fixed number of times

C) Indefinitely (loop never terminates)
D) Based on multiple conditions
Answer: A) Only once

2.3 Declaring Variables and Methods:
1. In Java, which keyword is used to declare a variable that cannot be modified once assigned?
A) final
B) static
C) const
D) var
Answer: A) final

2. Which access modifier allows a variable or method to be accessible only within its own class?
A) private
B) protected
C) public
D) package-private (default)
Answer: A) private

3. What is the syntax for declaring a method in Java?
A) `function methodName() {}`
B) `method methodName() {}`
C) `void methodName() {}`
D) `methodName() {}`
Answer: C) `void methodName() {}`

4. What is the return type of a method that does not return any value in Java?
A) void
B) int
C) String
D) boolean
Answer: A) void

5. In Java, a method can have _____ with the same name, but different parameter lists.
A) Overloaded methods
B) Nested methods
C)

Recursive methods
D) Abstract methods
Answer: A) Overloaded methods

2.4 Data Type Compatibility:
1. In Java, which type of conversion is performed automatically when a smaller data type is assigned to a larger data type?
A) Widening conversion
B) Narrowing conversion
C) Implicit conversion
D) Explicit conversion
Answer: A) Widening conversion

2. What happens if you try to assign a floating-point value to an integer variable in Java without explicit casting?
A) The program will not compile.
B) The decimal part of the value will be truncated.
C) The variable will be automatically promoted to a floating-point type.
D) The program will throw a runtime exception.
Answer: A) The program will not compile.

3. Which data type is used to store floating-point numbers with a higher precision than the `float` data type?
   A) double
   B) float128
   C) long double
   D) BigDecimal
   Answer: C) long double

4. What is the result of the following expression?
   `int result = 10 / 3;`
   A) 3.33333
   B) 3
   C) 3.0
   D) 3.33333...
   Answer: B) 3

5. In Java, which data type is used to store true/false values?
   A) boolean
   B) bool
   C) bit
   D) logical
   Answer: A) boolean

2.5 Get Yourself Acquainted with Java Environment. Print Different Patterns of Asterisk (*) Using Loops (e.g. Triangle of *).
1. To print a right-angled triangle of asterisks (*), how many nested loops are required?
   A) One
   B) Two
   C) Three
   D) It cannot be achieved using loops
   Answer: B) Two

2. What is the output of the following code snippet?
   ```java
   for (int i = 1; i <= 5; i++) {
       for (int j = 1; j <= i; j++) {
           System.out.print("* ");
       }
       System.out.println();
   }
   ```
   A) *
      * *
      * * *
      * * * *
      * * * * *
   B) * * * * *
      * * * *
      * * *
      * *
      *
   C) * * * * *
      * * * *
      * * *
      * *
      *
      * *
      * * *
      * * * *
      * * * * *

D) The code will not compile
Answer: A) *
* *
* * *
* * * *
* * * * *

3. To print a hollow square of asterisks (*), how many nested loops are required?
A) One
B) Two
C) Three
D) It cannot be achieved using loops
Answer: B) Two

4. What is the output of the following code snippet?
```java
int size = 5;
for (int i = 1; i <= size; i++) {
    for (int j = 1; j <= size; j++) {
        if (i == 1 || i == size || j == 1 || j == size) {
            System.out.print("* ");
        } else {
            System.out.print("  ");
        }
    }
    System.out.println();
}
```
A) * * * * *
* *
* *
* *
* * * * *
B) * * * * *
* *
* *
* *
* * * * *
C) * * * * *
* *
* *
* *
* * * * *
D) The code will not compile
Answer: A) * * * * *
* *
* *
* *
* * * * *

5. The pattern of asterisks printed using loops is primarily used for _____.
A) Debugging purposes
B) Beautifying the code
C) User input validation
D) Educational purposes
Answer: D) Educational purposes]

3. Lecture: Static Variables and Methods

3.1 Accessing Static Variables and Methods of Different Classes:

1. Static variables in Java are shared among _____.
   A) All instances of the class
   B) Only the current method
   C) All methods of the class
   D) Only the current instance of the class
   Answer: A) All instances of the class

2. Which keyword is used to access a static method of a different class in Java?
   A) this
   B) super
   C) static
   D) ClassName
   Answer: D) ClassName

3. In Java, you can access a static variable without creating an instance of the class because _____.
   A) Static variables are automatically initialized to their default values
   B) Static variables are always set to zero by default
   C) Static variables are shared across all instances of the class
   D) Static variables can only be accessed within the same class
   Answer: C) Static variables are shared across all instances of the class

4. What is the result of the following code snippet?
   ```java
   class MyClass {
       static int count = 0;

       MyClass() {
           count++;
       }
   }

   public class Main {
       public static void main(String[] args) {
           MyClass obj1 = new MyClass();
           MyClass obj2 = new MyClass();
           System.out.println(MyClass.count);
       }
   }
   ```
   A) 0
   B) 1
   C) 2
   D) The code will not compile
   Answer: C) 2

5. The keyword used to access a static method within the same class is _____.
   A) static
   B) this
   C) self
   D) ClassName
   Answer: D) ClassName

6. In Java, static variables are initialized _____.
   A) Automatically to their default values

B) Only when explicitly assigned a value
C) To null by default
D) In a random order
Answer: A) Automatically to their default values

7. Can static variables be accessed directly from an instance (object) of the class?
A) Yes, but only if the instance is not null
B) No, static variables can only be accessed using the class name
C) Yes, static variables can be accessed using both the instance and class name
D) Yes, but only if the static variable is public
Answer: C) Yes, static variables can be accessed using both the instance and class name

8. What is the difference between a static variable and an instance variable in Java?
A) Instance variables are shared among all instances of the class, while static variables are unique to each instance.
B) Instance variables can be accessed using the class name, while static variables can only be accessed using an instance of the class.
C) Static variables are created inside methods, while instance variables are created outside methods.
D) Instance variables are declared with the keyword "this," while static variables are declared with the keyword "static."
Answer: A) Instance variables are shared among all instances of the class, while static variables are unique to each instance.

3.2 Introduction to Reference Data Types:

1. Reference data types in Java store _____.
A) Primitive values like integers and booleans
B) Data references or memory addresses of objects
C) Literal values like strings and characters
D) Static variables and methods of classes
Answer: B) Data references or memory addresses of objects

2. Which of the following is a reference data type in Java?
A) int
B) double
C) boolean
D) String
Answer: D) String

3. In Java, reference data types are used to _____.
A) Represent simple data values
B) Store direct values like numbers and characters
C) Create complex data structures like arrays and objects
D) Define static variables and methods
Answer: C) Create complex data structures like arrays and objects

4. What is the default value of a reference variable in Java if it is not initialized?
A) 0
B) null
C) ""
D) false
Answer: B) null

5. Which of the following data types is a reference data type that allows multiple characters to be stored?
A) char
B) byte
C) int
D) String

Answer: D) String

6. Reference data types are stored in the _____ in Java.
A) Stack
B) Heap
C) Queue
D) Static memory
Answer: B) Heap

7. What is the key difference between a reference data type and a primitive data type in Java?
A) Reference data types are larger in size compared to primitive data types.
B) Reference data types are passed by value, whereas primitive data types are passed by reference.
C) Reference data types can store data directly, while primitive data types store memory addresses of data.
D) Reference data types represent complex objects, while primitive data types represent simple data values.
Answer: D) Reference data types represent complex objects, while primitive data types represent simple data values.

8. Which of the following is an example of a reference data type in Java?
A) double
B) float
C) Object
D) boolean
Answer: C) Object

3.3 Reference Variables and Methods:

1. Reference variables in Java are used to _____.
A) Store the actual data values
B) Store memory addresses of objects
C) Perform arithmetic calculations
D) Declare static variables
Answer: B) Store memory addresses of objects

2. What is the result of the following code snippet?
```java
class Person {
    String name;

    Person(String n) {
        name = n;
    }
}

public class Main {
    public static void main(String[] args) {
        Person person1 = new Person("Alice");
        Person person2 = new Person("Bob");
        person2 = person1;
        System.out.println(person2.name);
    }
}
```
A) Alice
B) Bob
C) null
D) The code will not compile
Answer: A) Alice

3. In Java, reference variables do not directly contain the _____.
A) Data values
B) Memory addresses of objects
C) Hashcode of objects
D) Reference type information
Answer: A) Data values

4. What happens when a reference variable is assigned to another reference variable in Java?
A) The data values of both variables are exchanged.
B) Both variables will point to the same memory address (object).
C) Java throws a compilation error.
D) The data values of the second variable are copied to the first variable.
Answer: B) Both variables will point to the same memory address (object).

5. In Java, reference variables can be used to access _____.
A) Static variables and methods

B) Primitive data types only
C) Only the object's hashcode
D) Methods of the corresponding class
Answer: D) Methods of the corresponding class

6. When a method is called on an object using a reference variable, the method that gets executed depends on _____.
A) Whether the method is static or non-static
B) The data type of the reference variable
C) The return type of the method
D) The actual type of the object being referred to
Answer: D) The actual type of the object being referred to

7. Which of the following statements about reference variables is true?
A) Reference variables are only used to access static members of a class.
B) Reference variables are always automatically initialized to null.
C) Reference variables can only refer to one object during their lifetime.
D) Reference variables are stored in the stack memory.
Answer: B) Reference variables are always automatically initialized to null.

8. Consider the following code:
```java
Person p1 = new Person("John");
Person p2 = p1;
```
What will be the value of `p2.name` if we change the value of `p1.name` to "Alice"?
A) "John"
B) "Alice"
C) null
D) Compilation error
Answer: B) "Alice"

3.4 Difference Between Reference Data Types and Primitive Data Types:

1. Which of the following is a difference between primitive data types and reference data types in Java?
A) Primitive data types are created on the heap, while reference data types are created on the stack.
B) Primitive data types store actual data values, while reference data types store memory addresses of objects.
C) Primitive data types can be used to create complex data structures, while reference data types are limited to simple data values.

D) Primitive data types can be null, while reference data types cannot be null.
Answer: B) Primitive data types store actual data values, while reference data types store memory addresses of objects.

2. Primitive data types in Java include _____.
A) String and Integer
B) int, float, char, and boolean
C) Object and Double
D) long, double, and boolean
Answer: B) int, float, char, and boolean

3. What is a key advantage of using primitive data types over reference data types?
A) Primitive data types have more memory storage than reference data types.
B) Primitive data types can be used to create complex data structures like arrays and objects.
C) Primitive data types are faster in terms of performance compared to reference data types.
D) Primitive data types can be null, while reference data types cannot be null.
Answer: C) Primitive data types are faster in terms of performance compared to reference data types.

4. Which of the following is true about the memory allocation of primitive data types and reference data types in Java?
A) Primitive data types are stored on the stack, while reference data types are stored on the heap.
B) Both primitive data types and reference data types are stored on the stack.
C) Primitive data types are stored on the heap, while reference data types are stored on the stack.
D) Both primitive data types and reference data types are stored on the heap.
Answer: A) Primitive data types are stored on the stack, while reference data types are stored on the heap.

5. Reference data types can have a value of _____.
A) 0
B) null
C) false
D) undefined
Answer: B) null

6. Which of the following data types is a reference data type in Java?
A) int
B) double
C) char
D) Integer
Answer: D) Integer

7. Primitive data types are _____ in Java.
A) Stored by reference
B) Immutable
C) Dynamic
D) Larger in size compared to reference data types
Answer: B) Immutable

8. In Java, reference data types are _____ in size compared to primitive data types.
A) Smaller
B) Larger
C) Equal
D) Unpredictable
Answer: B) Larger

3.5 Difference Between Reference Variable and Static Variable:

1. What is a key difference between a reference variable and a static variable in Java?

A) A reference variable is used to store memory addresses, while a static variable is used to store actual data values.
B) A reference variable can be accessed directly using the class name, while a static variable can only be accessed through an instance of the class.
C) A reference variable can be declared with the "static" keyword, while a static variable cannot be declared with the "static" keyword.
D) A reference variable can store primitive data types, while a static variable can store only reference data types.
Answer: A) A reference variable is used to store memory addresses, while a static variable is used to store actual data values.

2. Which of the following statements is true about reference variables in Java?
A) Reference variables are stored on the stack.
B) Reference variables can be declared with the "static" keyword.
C) Reference variables are shared across all instances of the class.
D) Reference variables can be accessed directly using the class name.
Answer: C) Reference variables are shared across all instances of the class.

3. A reference variable can refer to _____.
A) Only one object during its lifetime
B) Multiple objects during its lifetime
C) Only primitive data types
D) Only static variables
Answer: B) Multiple objects during its lifetime

4. What is the scope of a reference variable in Java?
A) The entire program
B) Only within the method where it is declared
C) Only within the class where it is declared
D) Only within the block where it is declared
Answer: D) Only within the block where it is declared

5. A static variable in Java is declared with the _____ keyword.
A) final
B) static
C) const
D) var
Answer: B) static

6. Static variables in Java are initialized _____.
A) Automatically to their default values
B) Only when explicitly assigned a value
C) To null by default
D) In a random order
Answer: A) Automatically to their default values

7. Which of the following is a true statement about static variables in Java?
A) Static variables are created on the stack.
B) Static variables can be accessed using the "this" keyword.
C) Static variables can only be accessed through an instance of the class.
D) Static variables are shared among all instances of the class.
Answer: D) Static variables are shared among all instances of the class.

8. A reference variable is used to _____.
A) Store memory addresses of objects
B) Store actual data values
C) Create complex data structures
D) Declare static methods
Answer: A) Store memory

4. Lecture: Constructors, Initializing Reference Variables Using Constructors

4.1 Pass by Value vs. Pass by Reference:

1. When a parameter is passed by value to a method, what happens to its value?
   a) The original value remains unchanged.
   b) The original value gets modified.
   c) The original value becomes null.
   d) The original value is deleted from memory.

   Answer: a) The original value remains unchanged.

2. In pass by reference, what is passed to a method?
   a) The value itself.
   b) A copy of the value.
   c) The memory address of the value.
   d) The data type of the value.

   Answer: c) The memory address of the value.

3. Which data types are always passed by value in Java?
   a) Primitive data types.
   b) Reference data types.
   c) Both primitive and reference data types.
   d) None of the above.

   Answer: a) Primitive data types.

4. When should you use pass by reference in Java?
   a) When you want to modify the original value inside the method.
   b) When you want to avoid modifying the original value.
   c) When dealing with primitive data types.
   d) Pass by reference is not supported in Java.

   Answer: a) When you want to modify the original value inside the method.

5. What is the default method of passing parameters in Java?
   a) Pass by value.
   b) Pass by reference.
   c) Pass by object.
   d) Pass by memory address.

   Answer: a) Pass by value.

6. Which of the following is true for pass by reference?
   a) It creates a copy of the object being passed.
   b) Any changes made to the parameter inside the method are reflected outside the method.
   c) It only applies to primitive data types.
   d) Java does not support pass by reference.

   Answer: b) Any changes made to the parameter inside the method are reflected outside the method.

7. Java uses pass by reference for all data types.
   a) True
   b) False

   Answer: b) False

4.2 Re-assigning a Reference Variable:

1. In Java, when you re-assign a reference variable to another object, what happens to the original object?
   a) The original object is deleted from memory.
   b) The original object remains unchanged.
   c) The original object becomes null.
   d) The original object becomes a copy of the new object.

   Answer: b) The original object remains unchanged.

2. What is the purpose of re-assigning a reference variable in Java?
   a) To create a copy of the original object.
   b) To delete the original object from memory.
   c) To make the original object eligible for garbage collection.
   d) To refer the variable to a different object.

   Answer: d) To refer the variable to a different object.

3. Which of the following is true about re-assigning reference variables?
   a) It is not allowed in Java.
   b) It is only allowed for primitive data types.
   c) It changes the type of the object being referred to.
   d) It has no impact on the original object.

   Answer: d) It has no impact on the original object.

4. In Java, when does an object become eligible for garbage collection after re-assigning its reference variable?
   a) Immediately after re-assigning the reference variable.
   b) When the new object is garbage collected.
   c) When there are no more references to the original object.
   d) The object is never eligible for garbage collection.

   Answer: c) When there are no more references to the original object.

5. Which statement is correct about re-assigning reference variables?
   a) It is done using the '->' operator.
   b) It can only be done in the main method.
   c) It changes the object's value, not the reference.
   d) It can lead to memory leaks if not done properly.

   Answer: d) It can lead to memory leaks if not done properly.

6. What is the result of re-assigning a reference variable to null?
   a) The object becomes unreachable and eligible for garbage collection.
   b) The object is immediately deleted from memory.
   c) The object's value becomes null.
   d) Java does not allow re-assigning reference variables to null.

   Answer: a) The object becomes unreachable and eligible for garbage collection.

7. Which data types can be re-assigned in Java?
   a) Only primitive data types.
   b) Only reference data types.
   c) Both primitive and reference data types.
   d) None of the above.

   Answer: c) Both primitive and reference data types.

4.3 Passing Reference Variable to Method:

1. When you pass a reference variable to a method in Java, what is passed to the method?
   a) The value of the object.
   b) A copy of the object.
   c) The memory address of the object.
   d) The reference variable itself.

   Answer: c) The memory address of the object.

2. What happens to the original object when a reference variable is passed to a method?
   a) The original object becomes a copy of the object inside the method.
   b) Any changes made to the object inside the method are reflected outside the method.
   c) The original object gets deleted from memory.
   d) The original object becomes null.

   Answer: b) Any changes made to the object inside the method are reflected outside the method.

3. What is the advantage of passing reference variables to a method in Java?
   a) It reduces the memory usage of the program.
   b) It allows the method to modify the original object's state.
   c) It speeds up the execution of the method.
   d) It prevents the original object from being garbage collected.

   Answer: b) It allows the method to modify the original object's state.

4. In Java, can a method modify the reference variable passed to it?
   a) Yes, a method can change the reference to point to a different object.
   b) No, Java does not allow modifying reference variables inside methods.
   c) Only primitive reference variables can be modified inside methods.
   d) It depends on the access level of the reference variable.

   Answer: a) Yes, a method can change the reference to point to a different object.

5. Which keyword is used in Java to pass an object by reference to a method?
   a) passbyref
   b) byref
   c) ref
   d) Java does not have pass by reference.

   Answer: d) Java does not have pass by reference.

6. What happens if a null reference variable is passed to a method in Java?
   a) It will cause a compilation error.
   b) The method will throw a NullPointerException.
   c) The method will create a new object with a null value.
   d) The method will ignore the null reference and continue execution.

   Answer: b) The method will throw a NullPointerException.

7. Which of the following is true regarding passing reference variables to a method?
   a) Java always passes reference variables by value.
   b) Java always passes reference variables by reference.
   c) The behavior depends on whether the variable is static or non-static.
   d) The behavior depends on the data type of the reference variable.

   Answer: a) Java always passes reference variables by value.

4.4 Initializing Reference Variable of Different Class:

1. In Java, how do you initialize a reference variable of a different class?
   a) Declare the variable and assign it a value using the "new" keyword.
   b) Use the "initialize" keyword followed by the class name.
   c) Declare the variable and set it equal to the class name.
   d) Use the "init" keyword followed by the class name.

   Answer: a) Declare the variable and assign it a value using the "new" keyword.

2. When initializing a reference variable of a different class, what is the type of object created?
   a) An instance of the class.
   b) A copy of the class.
   c) A subclass of the class.
   d) A superclass of the class.

   Answer: a) An instance of the class.

3. Which statement is true when initializing a reference variable?
   a) The variable is assigned a default value based on its data type.
   b) The variable points to an existing object in memory.
   c) The variable is automatically initialized to null.
   d) Java does not allow initializing reference variables of different classes.

   Answer: a) The variable is assigned a default value based on its data type.

4. What does the "new" keyword do when initializing a reference variable?
   a) It assigns the reference variable to a new object in memory.
   b) It creates a copy of the reference variable.
   c) It points the reference variable to a different memory location.
   d) The "new" keyword is not used for initializing reference variables.

   Answer: a) It assigns the reference variable to a new object in memory.

5. In Java, if you don't initialize a reference variable, what value does it have?
   a) null
   b) 0
   c) "undefined"
   d) The value depends on the data type of the variable.

   Answer: a) null

6. Which of the following is true when initializing reference variables of different classes?
   a) The reference variable can be directly assigned to a value of any data type.
   b) The reference variable must be initialized in the main method.
   c) The reference variable can only be initialized using a constructor.
   d) The reference variable must be declared as a static variable.

   Answer: c) The reference variable can only be initialized using a constructor.

7. What happens if you attempt to use an uninitialized reference variable?
   a) The Java compiler will automatically initialize it to a default value.
   b) The program will throw a compilation error.
   c) The program will terminate with a runtime exception.
   d) The reference variable will automatically point to an object with default values.

   Answer: b) The program will throw a compilation error.

4.5 Heap Memory and Stack Memory:

1. In Java, where are objects stored when they are dynamically allocated using the "new" keyword?
   a) Heap memory
   b) Stack memory
   c) Global memory
   d) Code memory

   Answer: a) Heap memory

2. What is the lifetime of objects stored in the heap memory?
   a) Objects exist as long as the program is running.
   b) Objects are automatically garbage collected after they go out of scope.
   c) Objects are stored permanently and cannot be removed.
   d) Objects are stored until the program is recompiled.

   Answer: a) Objects exist as long as the program is running.

3. In Java, where are local variables and method call data stored?
   a) Heap memory
   b) Stack memory
   c) Global memory
   d) Code memory

   Answer: b) Stack memory

4. Which memory area is used for storing references to objects in Java?
   a) Heap memory
   b) Stack memory
   c) Global memory
   d) Code memory

   Answer: b) Stack memory

5. How is memory management different for the heap and stack memory in Java?
   a) Heap memory is automatically managed by the Java Virtual Machine (JVM), while stack memory needs manual management.
   b) Stack memory is automatically managed by the JVM, while heap memory needs manual management.
   c) Both heap and stack memory require manual memory management.
   d) Java does not differentiate between heap and stack memory management.

   Answer: a) Heap memory is automatically managed by the Java Virtual Machine (JVM), while stack memory needs manual management.

6. Which memory area is responsible for keeping track of method call information, local variables, and partial results?
   a) Heap memory
   b) Stack memory
   c) Global memory
   d) Code memory

   Answer: b) Stack memory

7. In Java, what happens to objects stored in the heap memory once they are no longer referenced?
   a) They are automatically garbage collected to free up memory.
   b) They are moved to the stack memory for faster access.
   c) They become eligible for manual deletion by the programmer.
   d) They remain in the heap memory indefinitely.

   Answer: a) They are automatically garbage collected to free up memory.

Certainly! Here are the next set of MCQs:

4.6 Print Default Values of Static & Instance Variables for Different Data Types:

1. In Java, what is the default value of an uninitialized `int` instance variable?
   a) 0
   b) 1
   c) -1
   d) null

   Answer: a) 0

2. What is the default value of an uninitialized `double` static variable in Java?
   a) 0
   b) 1.0
   c) 0.0
   d) null

   Answer: c) 0.0

3. For a non-static object reference variable, what is the default value when not initialized?
   a) 0
   b) 1
   c) null
   d) "undefined"

   Answer: c) null

4. What is the default value of an uninitialized `boolean` instance variable?
   a) true
   b) false
   c) 0
   d) null

   Answer: b) false

5. In Java, what is the default value of an uninitialized `char` static variable?
   a) 'a'
   b) '0'
   c) '\u0000'
   d) null

   Answer: c) '\u0000'

6. Which of the following is true regarding default values in Java?
   a) All data types have the same default value, i.e., null.
   b) Default values are set based on the programmer's choice.
   c) Instance variables have default values, but static variables do not.
   d) Default values are automatically assigned by the JVM.

   Answer: d) Default values are automatically assigned by the JVM.

7. What happens if you try to use an uninitialized local variable in Java?
   a) The program will compile and run without any issues.
   b) The program will throw a compilation error.
   c) The program will throw a runtime exception.
   d) The variable will automatically be initialized to its default value.

   Answer: b) The program will throw a compilation error.

4.7 Build a Class Employee which Contains Details about the Employee and Compile and Run Its Instance:

1. What is the purpose of the Employee class in Java?
   a) To create a new Java file.
   b) To define the details and behavior of an employee.
   c) To store employee data in a database.
   d) To manage employee records in a file.

   Answer: b) To define the details and behavior of an employee.

2. Which access modifier should be used for the instance variables of the Employee class to restrict direct access from outside the class?
   a) public
   b) private
   c) protected
   d) default (package-private)

   Answer: b) private

3. How can you instantiate an object of the Employee class in Java?
   a) By using the `new` keyword followed by the class name and constructor parameters.
   b) By directly assigning values to the instance variables.
   c) By using the `init` keyword followed by the class name.
   d) By importing the Employee class from another package.

   Answer: a) By using the `new` keyword followed by the class name and constructor parameters.

4. Which method is called automatically when an object of the Employee class is instantiated?
   a) run()
   b) main()
   c) start()
   d) constructor

   Answer: d) constructor

5. To compile and run the Employee class, what should the Java file be named?
   a) Employee.java
   b) employee.class
   c) emp.class
   d) emp.java

   Answer: a) Employee.java

6. What is the purpose of the `main` method in the Employee class?
   a) To print the default values of the instance variables.
   b) To create and run an instance of the Employee class.
   c) To declare the instance variables of the Employee class.
   d) The `main` method is not required in the Employee class.

   Answer: b) To create and run an instance of the Employee class.

7. When creating an instance of the Employee class, what constructor parameters might you pass to set initial values for the instance variables?
   a) No parameters are needed; the constructor automatically initializes the instance variables.
   b) Employee name and age.
   c) Employee salary and department.
   d) Employee ID and job title.

   Answer: b) Employee name and age.

4.8 Build a Class which Has References to Other Classes. Instantiate these Reference Variables and Invoke Instance Methods:

1. In Java, how do you define a reference variable that refers to an object of another class?
   a) Declare the variable and assign it using the "new" keyword.
   b) Use the "ref" keyword followed by the class name.
   c) Declare the variable and assign it using the "instanceof" keyword.
   d) Java does not allow referencing objects of other classes.

   Answer: a) Declare the variable and assign it using the "new" keyword.

2. What is the purpose of having references to other classes in Java?
   a) To create new instances of the referenced classes.
   b) To access and utilize the methods and attributes of the referenced classes.
   c) To create a new data type in Java.
   d) To replace the existing class with the referenced class.

   Answer: b) To access and utilize the methods and attributes of the referenced classes.

3. How do you invoke an instance method on an object of another class?
   a) By using the "call" keyword followed by the method name.
   b) By declaring the method as public static in the referenced class.
   c) By using the reference variable followed by the method name and arguments.
   d) By importing the referenced class into the current class.

   Answer: c) By using the reference variable followed by the method name and arguments.

4. What is the role of the "dot" operator (.) in Java when invoking an instance method on a referenced object?
   a) It creates a new object from the referenced class.
   b) It calls the constructor of the referenced class.
   c) It accesses the methods and attributes of the referenced object.
   d) It deletes the referenced object from memory.

   Answer: c) It accesses the methods and attributes of the referenced object.

5. In Java, can a class reference multiple other classes simultaneously?
   a) Yes, a class can reference as many other classes as needed.
   b) No, a class can only reference one other class at a time.
   c) Java does not support referencing objects of other classes.
   d) It depends on the data types of the referenced classes.

   Answer: a) Yes, a class can reference as many other classes as needed.

6. What happens if you try to invoke a non-static method on a reference variable before initializing it with an object?
   a) The program will compile, but the method will not be executed.
   b) The program will throw a compilation error.
   c) The program will throw a runtime exception.
   d) The method will be executed, but it will return a null reference.

   Answer: c) The program will throw a runtime exception.

7. How do you initialize reference variables of other classes in Java?
   a) By using the "initialize" keyword followed by the class name.
   b) By directly assigning values to the reference variables.
   c) By using the "new" keyword followed by the class name and constructor parameters.
   d) Java automatically initializes reference variables to null.

Answer: c) By using the "new" keyword followed by the class name and constructor parameters.

5. Lecture: Introduction to OOP Concepts

Lecture: Introduction to OOP Concepts

1. Which of the following is not an object-oriented programming (OOP) concept?
   a) Inheritance
   b) Abstraction
   c) Modularity
   d) Encapsulation

   Answer: c) Modularity

2. In OOP, what is encapsulation?
   a) The process of combining data and methods into a single unit (object).
   b) The process of hiding the implementation details of an object's methods.
   c) The process of making a class inherit properties from another class.
   d) The process of creating multiple instances of a class.

   Answer: a) The process of combining data and methods into a single unit (object).

3. What is the main advantage of using OOP in software development?
   a) It reduces the complexity of the code.
   b) It eliminates the need for variables and loops.
   c) It allows for procedural programming.
   d) It optimizes memory usage.

   Answer: a) It reduces the complexity of the code.

4. Which concept in OOP allows you to create a new class based on an existing class?
   a) Polymorphism
   b) Inheritance
   c) Encapsulation
   d) Abstraction

   Answer: b) Inheritance

5. Which OOP concept promotes the reusability of code and establishes a hierarchical relationship between classes?
   a) Polymorphism
   b) Encapsulation
   c) Abstraction
   d) Inheritance

   Answer: d) Inheritance

6. What does the term "object" refer to in object-oriented programming?
   a) A specific instance of a class that encapsulates data and behavior.
   b) The main method of a Java program.
   c) A variable that holds a primitive data type value.
   d) The process of hiding implementation details.

   Answer: a) A specific instance of a class that encapsulates data and behavior.

7. Which OOP concept allows you to define a blueprint for creating objects?
   a) Inheritance
   b) Abstraction

c) Encapsulation
d) Polymorphism

Answer: b) Abstraction

8. Which of the following is true about classes in object-oriented programming?
a) Classes are instances of objects.
b) Classes represent behavior, while objects represent methods.
c) Classes define the data and behavior of objects.
d) Classes are only used in procedural programming.

Answer: c) Classes define the data and behavior of objects.

9. What is the primary purpose of a constructor in OOP?
a) To create multiple instances of a class.
b) To provide a blueprint for objects.
c) To access private data members of a class.
d) To initialize the state of an object.

Answer: d) To initialize the state of an object.

10. Which OOP concept allows a class to have multiple methods with the same name but different parameter lists?
a) Encapsulation
b) Abstraction
c) Polymorphism
d) Inheritance

Answer: c) Polymorphism

11. What is the access specifier that allows a class member to be accessible only within the same package?
a) private
b) protected
c) public
d) default (package-private)

Answer: d) default (package-private)

12. Which of the following is NOT a fundamental principle of OOP?
a) Inheritance
b) Encapsulation
c) Polymorphism
d) Complexity

Answer: d) Complexity

13. What is the term used to describe the ability of an object to take on many forms?
a) Abstraction
b) Encapsulation
c) Polymorphism
d) Inheritance

Answer: c) Polymorphism

14. In OOP, what is the purpose of the "super" keyword in Java?
a) To create a new instance of a superclass.
b) To access the superclass's constructor or methods.
c) To specify multiple inheritance in Java.
d) To override the superclass's methods.

Answer: b) To access the superclass's constructor or methods.

15. Which OOP concept allows you to define a new class based on an existing class but with some modifications or extensions?
a) Abstraction
b) Polymorphism
c) Inheritance
d) Encapsulation

Answer: c) Inheritance

5.1 Encapsulation

1. In encapsulation, how are the data members of a class accessed?
a) Directly using the dot (.) operator.
b) Through getter and setter methods.
c) By declaring them as public.
d) By importing the class in another package.

Answer: b) Through getter and setter methods.

2. Which access specifier is commonly used for the getter and setter methods in encapsulation?
a) private
b) protected
c) public
d) default (package-private)

Answer: c) public

3. What is the main benefit of encapsulation in OOP?
a) Improved code reusability.
b) Simplified debugging process.
c) Enhanced code readability.
d) Data hiding and protection.

Answer: d) Data hiding and protection.

4. In encapsulation, why do we declare data members as private?
a) To make them accessible from any class.
b) To prevent direct access from outside the class.
c) To allow subclasses to access them.
d) To avoid using getter and setter methods.

Answer: b) To prevent direct access from outside the class.

5. How do getter and setter methods promote encapsulation?
a) They enforce strict encapsulation rules.
b) They provide a convenient way to access and modify the data members.
c) They automatically initialize data members with default values.
d) They prevent other methods from accessing the data members.

Answer: b) They provide a convenient way to access and modify the data members.

6. In encapsulation, which method is used to read the value of a private data member?
a) readValue()
b) getValue()
c) getData()
d) getVariableName()

Answer: b) getValue()

7. What is the purpose of using the "this" keyword in getter and setter methods?
a) To refer to the superclass.
b) To call the constructor of the class.
c) To refer to the current object.
d) To invoke static methods.

Answer: c) To refer to the current object.

8. How does encapsulation help in maintaining code flexibility?
a) It allows access to all data members from any class.
b) It enforces strict inheritance rules.
c) It isolates the implementation details of a class.
d) It automatically initializes variables to default values.

Answer: c) It isolates the implementation details of a class.

9. What happens if you bypass the getter and setter methods and directly access the private data members?
a) It will

result in a compilation error.
b) It will not affect the encapsulation of the class.
c) It will lead to improved performance.
d) It will violate the encapsulation principle.

Answer: d) It will violate the encapsulation principle.

10. In encapsulation, how do you provide write-only access to a private data member?
a) By declaring the data member as public.
b) By providing a public setter method but no getter method.
c) By providing a public getter method but no setter method.
d) Java does not allow write-only access to private data members.

Answer: b) By providing a public setter method but no getter method.

11. Which of the following is true about encapsulation?
a) Encapsulation increases the complexity of the code.
b) Encapsulation makes it harder to maintain and debug the code.
c) Encapsulation allows you to hide the internal implementation details of a class.
d) Encapsulation is only applicable to primitive data types.

Answer: c) Encapsulation allows you to hide the internal implementation details of a class.

12. What is the scope of a private data member in a class that implements encapsulation?
a) It can be accessed from any class.
b) It can be accessed from subclasses only.
c) It can be accessed only within the same class.
d) It can be accessed from any class in the same package.

Answer: c) It can be accessed only within the same class.

13. In encapsulation, what is the purpose of using the "final" keyword with a data member?
a) To prevent the data member from being accessed.
b) To make the data member read-only (constant).
c) To make the data member invisible to other classes.
d) To make the data member accessible without getter and setter methods.

Answer: b) To make the data member read-only (constant).

14. How can you ensure that the value of a data member remains within a specific range in encapsulation?
a) By declaring the data member as public.
b) By declaring the data member as static.
c) By using conditional statements in the getter and setter methods.
d) Java does not allow setting constraints on data members.

Answer: c) By using conditional statements in the getter and setter methods.

15. In encapsulation, what is the role of access modifiers (e.g., private, public) for methods and data members?
a) They define the data types for methods and data members.
b) They determine the order of execution for methods.
c) They control the visibility and accessibility of methods and data members.
d) They enforce encapsulation rules automatically.

Answer: c) They control the visibility and accessibility of methods and data members.

5.2 Inheritance: Single & Multilevel

1. What is inheritance in OOP?
a) The process of hiding the implementation details of a class.
b) The process of creating multiple instances of a class.
c) The process of acquiring the properties and behaviors of an existing class.
d) The process of converting an object to a different data type.

Answer: c) The process of acquiring the properties and behaviors of an existing class.

2. What is the base class (or superclass) in inheritance?
a) The class that inherits properties and behaviors from another class.
b) The class that is derived from another class.
c) The class that contains only private data members.
d) The class that is at the top of the class hierarchy.

Answer: d) The class that is at the top of the class hierarchy.

3. Which keyword is used in Java to achieve inheritance?
a) extend
b) implement
c) inherit
d) derive

Answer: a) extend

4. In single inheritance, a subclass can inherit from _____.
a) multiple superclasses
b) only one superclass
c) multiple subclasses
d) any class in the same package

Answer: b) only one superclass

5. What is the relationship between the subclass and the superclass in inheritance?
a) The subclass is a specialization of the superclass.
b) The subclass is an exact copy of the superclass.
c) The subclass is a separate entity unrelated to the superclass.
d) The subclass is a combination of multiple superclasses.

Answer: a) The subclass is a specialization of the superclass.

6. What is the main advantage of using inheritance in OOP?
   a) It allows for multiple inheritance, reducing code duplication.
   b) It allows for reusing code from existing classes.
   c) It eliminates the need for constructors in subclasses.
   d) It simplifies the process of creating new classes.

   Answer: b) It allows for reusing code from existing classes.

7. In inheritance, what is a method override?
   a) A method that is declared as private in the superclass.
   b) A method that is accessible only from the subclass.
   c) A method that has the same name and signature in the subclass as in the superclass.
   d) A method that is declared as final in the superclass.

   Answer: c) A method that has the same name and signature in the subclass as in the superclass.

8. Which access modifier is commonly used for the methods in a base class to allow subclasses to override them?
   a) private
   b) protected
   c) public
   d) default (package-private)

   Answer: b) protected

9. What is the purpose of the "super" keyword in inheritance?
   a) To call the constructor of the subclass.
   b) To create a new instance of the superclass.
   c) To call the constructor of the superclass.
   d) To override a method in the superclass.

   Answer: c) To call the constructor of the superclass.

10. In Java, can a subclass inherit private members (data members and methods) from the superclass?
   a) Yes, a subclass can inherit private members.
   b) No, private members are not inherited by subclasses.
   c) It depends on the access modifier used for the private members.
   d) It depends on whether the superclass is abstract.

   Answer: b) No, private members are not inherited by subclasses.

11. What happens if a subclass tries to access a private member (data member or method) of its superclass?
   a) The program will compile but throw a runtime exception.
   b) The program will throw a compilation error.
   c) The private member will be accessible from the subclass.
   d) The superclass will automatically become a subclass.

   Answer: b) The program will throw a compilation error.

12. In multilevel inheritance, a subclass can inherit from _____.
   a) multiple superclasses
   b) only one superclass
   c) multiple subclasses
   d) a subclass of another subclass

   Answer: d) a subclass of another subclass

13. What is the top-level superclass in multilevel inheritance?

a) The class that is derived from all other classes.
b) The class that is at the top of the class hierarchy.
c) The class that contains only private data members.
d) The class that inherits from multiple superclasses

.

Answer: b) The class that is at the top of the class hierarchy.

14. In multilevel inheritance, how many levels of inheritance can a subclass have?
a) Only one level
b) Two levels
c) Three levels
d) Any number of levels

Answer: d) Any number of levels

15. What is the potential disadvantage of using multilevel inheritance?
a) Increased code complexity and reduced reusability.
b) Inability to override methods in subclasses.
c) Difficulty in accessing superclass members.
d) Inability to have more than one superclass.

Answer: a) Increased code complexity and reduced reusability.

6. Lecture: Inheritance: Hierarchical

6.Inheritance: Hierarchical

1. In hierarchical inheritance, how many classes are derived from a single base class?
a) One
b) Two
c) Multiple
d) None

Answer: b) Two

2. Which keyword is used to implement inheritance in Java?
a) this
b) new
c) super
d) extends

Answer: d) extends

3. What is the main advantage of hierarchical inheritance?
a) It allows a class to inherit from multiple base classes.
b) It enables code reusability and reduces redundancy.
c) It ensures that all classes have the same instance variables.
d) It supports multiple constructors for a class.

Answer: b) It enables code reusability and reduces redundancy.

4. Which type of inheritance is represented by the following code snippet?

```java
class A {
```

```
    // ...
}

class B extends A {
    // ...
}

class C extends A {
    // ...
}
```

a) Single inheritance
b) Hierarchical inheritance
c) Multiple inheritance
d) Multilevel inheritance

Answer: b) Hierarchical inheritance

5. Can a subclass access private members of its superclass in Java?
a) Yes, directly without any restrictions
b) Yes, but only using the "super" keyword
c) No, it cannot access private members
d) Only if the subclass is in the same package as the superclass

Answer: c) No, it cannot access private members

6. What is the main purpose of the "protected" access modifier in Java?
a) To allow unrestricted access to class members
b) To restrict access only to the class itself
c) To allow access to class members within the same package and subclasses
d) To make class members accessible globally

Answer: c) To allow access to class members within the same package and subclasses

7. Which keyword is used to prevent a class from being inherited in Java?
a) abstract
b) static
c) final
d) private

Answer: c) final

8. What is the process called when a subclass provides a specific implementation for a method that is already defined in its superclass?
a) Overriding
b) Overloading
c) Inheriting
d) Encapsulating

Answer: a) Overriding

6.1 Polymorphism: Compile Time and Runtime Polymorphism

1. What is polymorphism in Java?
a) The ability of a class to inherit properties from multiple base classes
b) The process of converting a primitive data type into an object
c) The ability of a method to take on different forms based on the number or type of its parameters
d) The process of creating multiple instances of a class

Answer: c) The ability of a method to take on different forms based on the number or type of its parameters

2. What is compile-time polymorphism also known as in Java?
   a) Method overloading
   b) Method overriding
   c) Method hiding
   d) Method wrapping

Answer: a) Method overloading

3. Which of the following statements is true about method overloading in Java?
   a) The return type of the method must be different for each overloaded version.
   b) Overloaded methods must have different method names.
   c) Overloaded methods must have the same number and types of parameters.
   d) Method overloading is not allowed in Java.

Answer: c) Overloaded methods must have the same number and types of parameters.

4. What is runtime polymorphism also known as in Java?
   a) Method overloading
   b) Method overriding
   c) Method hiding
   d) Method wrapping

Answer: b) Method overriding

5. Which keyword is used to annotate a method in the superclass that can be overridden by its subclasses?
   a) this
   b) abstract
   c) super
   d) override

Answer: d) override

6. In Java, when does the determination of which method to call happen in the case of method overloading?
   a) At compile-time
   b) At runtime
   c) At the time of object creation
   d) When the program is executed

Answer: a) At compile-time

7. Which type of polymorphism is resolved at compile time in Java?
   a) Runtime polymorphism
   b) Compile-time polymorphism
   c) Dynamic polymorphism
   d) Static polymorphism

Answer: b) Compile-time polymorphism

8. Consider the following method:

```java
public void display(int a, double b) {
    // Method implementation
}
```

Which of the following method calls would be considered as an example of method overloading?

a) display(10, 5.0)
b) display(5.0, 10)
c) display(10, 10)
d) display(10, 10, 5.0)

Answer: a) display(10, 5.0)

6.2 Rules of Overriding and Overloading of Methods

1. Which of the following is NOT a rule for method overriding in Java?
a) The access level of the overriding method cannot be more restrictive than the overridden method.
b) The return type of the overriding method must be the same as the overridden method, or a subtype of it.
c) The overriding method can throw any checked exceptions, regardless of the exceptions thrown by the overridden method.
d) The static methods can be overridden in Java.

Answer: d) The static methods can be overridden in Java.

2. What happens if a subclass tries to override a method in its superclass that is marked as "final"?
a) The subclass can override the final method without any issues.
b) The compiler will generate an error, and the method cannot be overridden.
c) The subclass can only override the method if it is also marked as "final."
d) The behavior is undefined, and it depends on the specific Java compiler.

Answer: b) The compiler will generate an error, and the method cannot be overridden.

3. In method overriding, what should be the relationship between the access modifiers of the overridden method and the overriding method?
a) The access modifier of the overriding method should be less restrictive than that of the overridden method.
b) The access modifier of the overriding method should be more restrictive than that of the overridden method.
c) The access modifiers of both methods should be the same.
d) The access modifiers are unrelated to method overriding.

Answer: c) The access modifiers of both methods should be the same.

4. Which annotation is used to indicate that a method is intended to override a superclass method in Java?
a) @override
b) @Override
c) @extend
d) @super

Answer: b) @Override

5. What is the result of method overloading in Java?
a) A single method can have multiple implementations with the same name.
b) A method in the subclass completely replaces the method in the superclass.
c) The compiler

automatically generates new methods based on the parameters provided.
d) The method with the same name and parameters in the superclass is hidden in the subclass.

Answer: a) A single method can have multiple implementations with the same name.

6. Consider the following code:

```java
class Animal {
    public void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    public void makeSound() {
        System.out.println("Dog barks");
    }
}
```

What is the result of calling `makeSound()` on an object of the `Dog` class?

a) It will print "Animal makes a sound."
b) It will print "Dog barks."
c) It will print both "Animal makes a sound" and "Dog barks."
d) It will result in a compilation error.

Answer: b) It will print "Dog barks."

7. Which of the following is true about method overloading in Java?
a) Overloaded methods must have different return types.
b) Overloaded methods must have different access modifiers.
c) Overloaded methods must have the same method name but different parameters.
d) Overloaded methods must belong to different classes.

Answer: c) Overloaded methods must have the same method name but different parameters.

8. In method overloading, can two methods have the same name and parameters but different return types?
a) Yes, it is allowed in Java.
b) No, it will result in a compilation error.
c) Only if the methods are declared in different classes.
d) Only if the methods have different access modifiers.

Answer: b) No, it will result in a compilation error.

6.3 super and this Keywords

1. In Java, what does the "super" keyword refer to?
a) It refers to the superclass and is used to call its methods or access its properties.
b) It refers to the current class and is used to call its methods or access its properties.
c) It refers to an object of the class and is used to create new instances.
d) It is used to specify the visibility of a class or a method.

Answer: a) It refers to the superclass and is used to call its methods or access its properties.

2. What is the purpose of the "this" keyword in Java?
a) It is used to create a copy of an object.
b) It is used to refer to the superclass.
c) It is used to call a method within another method.
d) It is used to refer to the current instance of the class.

Answer: d) It is used to refer to the current instance of the class.

3. Consider the following code snippet:

```java
class Parent {
    int x;

    Parent(int x) {
        this.x = x;
    }
}

class Child extends Parent {
    int y;

    Child(int x, int y) {
        super(x);
        this.y = y;
    }
}
```

What is the purpose of the "super(x)" statement in the Child class constructor?

a) It calls the default constructor of the Parent class.
b) It sets the value of x in the Parent class to the value passed in the Child class constructor.
c) It creates a new object of the Parent class.
d) It is not a valid statement in Java.

Answer: b) It sets the value of x in the Parent class to the value passed in the Child class constructor.

4. In which of the following scenarios would you use the "super" keyword in Java?
a) To create a new instance of a class.
b) To call a method in the current class.
c) To call a constructor of the superclass from a subclass constructor.
d) To define a new method in a subclass.

Answer: c) To call a constructor of the superclass from a subclass constructor.

5. What happens if the "super" keyword is not used in a subclass constructor in Java?
a) The subclass constructor will automatically call the default constructor of the superclass.
b) The compiler will generate an error, and the subclass constructor won't compile.
c) The subclass constructor will throw a runtime exception.
d) The subclass constructor will have no effect, and it won't be able to create instances of the subclass.

Answer: a) The subclass constructor will automatically call the default constructor of the superclass.

6. In Java, is it possible to use both "this" and "super" in the same constructor?
a) Yes, but only if the superclass has a parameterless constructor.
b) No, using both "this" and "super" in the same constructor is not allowed.
c) Yes, using "this" and "super" together is common when chaining constructors in a class.
d) Yes, but only if the superclass has a constructor with the same number of parameters as the subclass.

Answer: c) Yes, using "this" and "super" together is common when chaining constructors in a class.

7. Which keyword is used to invoke a method of the superclass in a subclass, even if the subclass has overridden that method?
a) override
b) overload
c) super

d) this

Answer: c) super

8. In Java, which keyword is used to differentiate between an instance variable of a class and a parameter in a method with the same name?
a) final
b) new
c) this
d) super

Answer: c) this

6.4 Upcasting & Downcasting of a Reference Variable

1. In Java, what is upcasting?
a) Converting a primitive data type into an object
b) Converting an object of a subclass to an object of the superclass
c) Converting an object of a superclass to an object of the subclass
d) Converting a reference variable to a primitive data type

Answer: b) Converting an object of a subclass to an object of the superclass

2. Which of the following statements is true about upcasting in Java?
a) Upcasting is always allowed and does not require any explicit casting.
b) Upcasting is only allowed if the superclass is abstract.
c) Upcasting may result in a loss of subclass-specific features and methods.
d) Upcasting is not supported in Java.

Answer: a) Upcasting is always allowed and does not require any explicit casting.

3. Consider the following code:

```java
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Dog barks");
    }
}
```

What is the result of the following code snippet?

```java
Animal animal = new Dog();
animal.makeSound();
```

a) It will print "Animal makes a sound."
b) It will print "Dog barks."
c) It will result in a compilation error.
d) It will throw a runtime exception.

Answer: b) It will print "Dog barks."

4. What is downcasting in Java?
   a) Converting a primitive data type into an object
   b) Converting an object of a subclass to an object of the superclass
   c) Converting an object of a superclass to an object of the subclass
   d) Converting a reference variable to a primitive data type

Answer: c) Converting an object of a superclass to an object of the subclass

5. Which of the following statements is true about downcasting in Java?
   a) Downcasting requires explicit casting using the "downcast" keyword.
   b) Downcasting can result in a loss of data if the object is not of the expected subclass type.
   c) Downcasting can only be performed between unrelated classes.
   d) Downcasting is an automatic process and doesn't require any special handling.

Answer: b) Downcasting can result in a loss of data if the object is not of the expected subclass type.

6. Consider the following code:

```java
class Animal {
    void makeSound() {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    void makeSound() {
        System.out.println("Cat meows");
    }
}
```

What will happen when you try to compile and run the following code snippet?

```java
Animal animal = new Dog();
Cat cat = (Cat) animal;
cat.makeSound();
```

   a) It will print "Animal makes a sound."
   b) It will print "Dog barks."
   c) It will print "Cat meows."
   d) It will result in a compilation error or a ClassCastException at runtime.

Answer: d) It will result in a compilation error or a ClassCastException at runtime.

7. To avoid the risk of ClassCastException when performing downcasting in Java, you should use which keyword?
   a) instanceof
   b) checkcast
   c) safecast

d) convert

Answer: a) instanceof

8. What does the "instanceof" keyword do in Java?
a) Checks if an object is an instance of the "Object" class.
b) Checks if an object is an instance of the superclass.
c) Checks if an object is an instance of a specific class or its subclasses.
d) Checks if an object has been initialized.

Answer: c) Checks if an object is an instance of a specific class or its subclasses.

6.5 Create a Class Employee and Encapsulate the Data Members. Create Demo Applications to Illustrate Different Types of Inheritance.

1. What is encapsulation in Java?
a) A mechanism to combine data and functions into a single unit.
b) The process of creating multiple instances of a class.
c) The process of hiding the internal details of a class and providing access through methods.
d) Storing data in different variables based on the data type.

Answer: c) The process of hiding the internal details of a class and providing access through methods.

2. Why is encapsulation important in Java?
a) It improves code readability.
b) It prevents unauthorized access to the data members of a class.
c) It eliminates the need for using objects in Java.
d) It enables the use of multiple inheritance in Java.

Answer: b) It prevents unauthorized access to the data members of a class.

3. Create a Java class named "Employee" with private data members "name," "id," and "salary." What is the correct way to provide access to these private data members?

a) Define public methods to get and set the data members' values.
b) Declare the data members as public to allow direct access.
c) Use the "protected" access modifier to make the data members accessible to subclasses.
d) Use the "this" keyword to access the data members directly.

Answer: a) Define public methods to get and set the data members' values.

4. How does encapsulation help in maintaining data integrity in a class?
a) By allowing direct access to data members, ensuring data consistency.
b) By preventing direct access to data members and providing controlled access through methods.
c) By making the data members public, enabling external classes to modify them directly.
d) By automatically initializing the data members with default values.

Answer: b) By preventing direct access to data members and providing controlled access through methods.

5. How can you achieve multiple inheritance in Java?
a) By defining a class with multiple parent classes using the "extends" keyword.
b) By implementing multiple interfaces in a single class.
c) Java does not support multiple inheritance.
d) By creating subclasses from multiple superclasses using the "super" keyword.

Answer: c) Java does not support multiple inheritance.

6. Create a Java class named "Manager" that inherits from the "Employee" class. Which keyword is used to achieve inheritance in Java?

a) extends
b) implements
c) super
d) this

Answer: a) extends

7. How can you access a method defined in the superclass from a subclass in Java?
a) By redefining the method with the same name in the subclass.
b) By using the "this" keyword to call the method.
c) By using the "super" keyword followed by the method name.
d) By casting the subclass object to the superclass type.

Answer: c) By using the "super" keyword followed by the method name.

8. In Java, what is the default access modifier for a class member if no access modifier is specified

?
a) public
b) private
c) protected
d) package-private (default)

Answer: d) package-private (default)

7. Lecture: Abstract Class and Abstract Methods

1. What is an abstract class in Java?
a) A class that cannot be instantiated and serves as a blueprint for other classes.
b) A class that can only contain static methods and variables.
c) A class that has no methods or variables defined in it.
d) A class that can only be used as a subclass and cannot have any subclasses.

Answer: a) A class that cannot be instantiated and serves as a blueprint for other classes.

2. Which keyword is used to define an abstract class in Java?
a) abstract
b) class
c) interface
d) implements

Answer: a) abstract

3. Can an abstract class have non-abstract (concrete) methods in Java?
a) Yes, an abstract class can only have concrete methods.
b) No, an abstract class can only have abstract methods.
c) Yes, an abstract class can have both abstract and concrete methods.
d) No, an abstract class cannot have any methods.

Answer: c) Yes, an abstract class can have both abstract and concrete methods.

4. What is the purpose of defining an abstract method in an abstract class?
a) To force the subclass to implement the abstract method.
b) To make the method inaccessible from any subclass.
c) To ensure that the method is implemented only in the abstract class.
d) To provide a default implementation of the method.

Answer: a) To force the subclass to implement the abstract method.

5. Can an abstract class have a constructor in Java?
   a) Yes, but the constructor must be private.
   b) Yes, and it must be public.
   c) No, an abstract class cannot have a constructor.
   d) Yes, but it must be protected.

Answer: b) Yes, and it must be public.

6. Which of the following statements about abstract classes is true?
   a) An abstract class can be directly instantiated using the "new" keyword.
   b) An abstract class must implement all the methods of its superclass.
   c) An abstract class can have both static and non-static methods.
   d) An abstract class cannot have any variables.

Answer: c) An abstract class can have both static and non-static methods.

7. Can a class be both abstract and final in Java?
   a) Yes, but it can only contain static methods and variables.
   b) Yes, but it cannot have any methods or variables.
   c) No, a class cannot be both abstract and final at the same time.
   d) Yes, and it can only have private methods.

Answer: c) No, a class cannot be both abstract and final at the same time.

8. If a class extends an abstract class, what are the options regarding the abstract methods in the subclass?
   a) The subclass can choose to implement all, some, or none of the abstract methods.
   b) The subclass must override all the abstract methods to provide implementations.
   c) The subclass can only override the non-abstract methods from the abstract class.
   d) The subclass inherits the concrete methods and cannot override the abstract methods.

Answer: a) The subclass can choose to implement all, some, or none of the abstract methods.

9. What happens if a non-abstract subclass fails to implement all the abstract methods from its abstract superclass?
   a) The subclass will inherit the abstract methods and become abstract itself.
   b) The subclass will have a default implementation of the missing abstract methods.
   c) The code will not compile, resulting in a compilation error.
   d) The abstract methods will be automatically inherited from the Object class.

Answer: c) The code will not compile, resulting in a compilation error.

10. Can an abstract class have variables with different access modifiers (e.g., public, private, protected) in Java?
   a) Yes, but only if the variables are declared as final.
   b) Yes, there are no restrictions on the access modifiers for variables in an abstract class.
   c) No, all variables in an abstract class must have the same access modifier.
   d) No, an abstract class cannot have any variables.

Answer: b) Yes, there are no restrictions on the access modifiers for variables in an abstract class.

11. In Java, can an abstract class extend another abstract class?
   a) Yes, and the subclass must implement all the abstract methods from both abstract classes.
   b) Yes, and the subclass can choose which abstract methods to implement from each abstract class.
   c) No, Java does not allow an abstract class to extend another abstract class.
   d) No, an abstract class can only extend a concrete class.

Answer: a) Yes, and the subclass must implement all the abstract methods from both abstract classes.

12. What is the purpose of making an abstract class?
   a) To prevent any instances of the class from being created.
   b) To allow instances of the class to be created without any restrictions.
   c) To ensure that the class cannot be subclassed.
   d) To hide the implementation details of the class.

Answer: a) To prevent any instances of the class from being created.

13. In Java, can a subclass inherit from multiple abstract classes?
   a) Yes, Java supports multiple inheritance from abstract classes.
   b) Yes, but the subclass must override all the methods from each abstract class.
   c) No, Java does not support multiple inheritance from abstract classes.
   d) Yes, but the subclass can only inherit from two abstract classes at most.

Answer: c) No, Java does not support multiple inheritance from abstract classes.

14. If a subclass inherits from both an abstract class and a concrete class, in which order should the "extends" keyword be used in Java?
   a) The order does not matter; the "extends" keyword can be used in any order.
   b) The subclass must extend the concrete class first, followed by the abstract class.
   c) The subclass must extend the abstract class first, followed by the concrete class.
   d) The subclass cannot inherit from both an abstract class and a concrete class.

Answer: c) The subclass must extend the abstract class first, followed by the concrete class.

15. Can a method be both abstract and static in Java?
   a) Yes, but only if the abstract method is private.
   b) Yes, but only if the abstract method is public.
   c) No, an abstract method cannot be static.
   d) Yes, but only if the abstract method is final.

Answer: c) No, an abstract method cannot be static.

Coding MCQs

1. Consider the following abstract class:

```java
abstract class Shape {
   int sides;

   abstract void draw();

   void displaySides() {
      System.out.println("Number of sides: " + sides);
   }
}
```

Which of the following subclass definitions is correct?

a)
```java
class Circle extends Shape {
   void draw() {
      System.out.println("Drawing a circle.");
   }
}
```

b)
```java
class Triangle extends Shape {
    void draw() {
        System.out.println("Drawing a triangle.");
    }
    void displaySides() {
        System.out.println("Number of

 sides: 3");
    }
}
```

c)
```java
class Square extends Shape {
    void displaySides() {
        System.out.println("Number of sides: 4");
    }
}
```

d)
```java
class Rectangle extends Shape {
    void draw() {
        System.out.println("Drawing a rectangle.");
    }
}
```

Answer: a)
```java
class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle.");
    }
}
```

2. What will be the output of the following code?

```java
abstract class Animal {
    abstract void makeSound();
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Dog barks.");
    }
}

class Cat extends Animal {
    void makeSound() {
        System.out.println("Cat meows.");
    }
}
```

```
public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog();
        Animal cat = new Cat();
        dog.makeSound();
        cat.makeSound();
    }
}
```

a) "Dog barks. Cat meows."
b) "Dog barks. Dog barks."
c) "Cat meows. Cat meows."
d) Compilation error due to abstract methods in the Animal class.

Answer: a) "Dog barks. Cat meows."

3. Which of the following is a valid way to declare and initialize an array of abstract class type in Java?

a)
```java
Animal[] animals = new Animal[];
```

b)
```java
Animal[] animals = new Animal[5];
```

c)
```java
Animal[] animals = { new Dog(), new Cat() };
```

d)
```java
Animal[] animals = new Animal[5] { new Dog(), new Cat() };
```

Answer: b)
```java
Animal[] animals = new Animal[5];
```

4. Consider the following code:

```java
abstract class Vehicle {
    abstract void start();
}

class Car extends Vehicle {
    void start() {
        System.out.println("Car started.");
    }
}

class Bike extends Vehicle {
    void start() {
```

```
        System.out.println("Bike started.");
    }
}

public class Main {
    public static void main(String[] args) {
        Vehicle vehicle = new Car();
        vehicle.start();
    }
}
```

What will be the output when the Main class is executed?

a) "Car started."
b) "Bike started."
c) Compilation error due to abstract methods in the Vehicle class.
d) Runtime error due to the abstract start() method not being implemented.

Answer: a) "Car started."

5. Can an abstract class have a non-abstract method?

a) Yes, but it must have only one non-abstract method.
b) Yes, and it can have any number of non-abstract methods.
c) No, an abstract class can only have abstract methods.
d) No, an abstract class cannot have any methods.

Answer: b) Yes, and it can have any number of non-abstract methods.

6. Consider the following code:

```java
abstract class Fruit {
    String color;

    Fruit(String color) {
        this.color = color;
    }

    abstract void eat();
}

class Apple extends Fruit {
    Apple(String color) {
        super(color);
    }

    void eat() {
        System.out.println("Eating an apple.");
    }
}
```

Which of the following is the correct way to create an instance of the Apple class?

a)
```java
Fruit apple = new Fruit("red");
```

b)
```java
Apple apple = new Apple("red");
```

c)
```java
Fruit apple = new Apple("red");
```

d)
```java
Apple apple = new Fruit("red");
```

Answer: c)
```java
Fruit apple = new Apple("red");
```

7. Can an abstract class be declared as final in Java?

a) Yes, and it can be instantiated directly using the "new" keyword.
b) Yes, but it cannot have any abstract methods.
c) No, an abstract class cannot be declared as final.
d) Yes, but it cannot have any concrete (non-abstract) methods.

Answer: c) No, an abstract class cannot be declared as final.

8. Consider the following code:

```java
abstract class Shape {
    abstract void draw();

    void display() {
        System.out.println("Displaying shape.");
    }
}

class Circle extends Shape {
    void draw() {
        System.out.println("Drawing a circle.");
    }
}
```

What will be the output of the following code?

```java
Shape shape = new Circle();
shape.draw();
shape.display();
```

a) "Drawing a circle. Displaying shape."
b) "Displaying shape. Drawing a circle."
c) Compilation error due to the abstract methods in the Shape class.
d) Runtime error due to the abstract draw() method not being implemented.

Answer: a) "Drawing a circle. Displaying shape."

9. In Java, can a concrete (non-abstract) class extend an abstract class?

a) Yes, but the concrete class must override all the abstract methods from the abstract class.
b) Yes, and the concrete class inherits the implementation of the abstract methods from the abstract class.
c) No, a concrete class cannot extend an abstract class.
d) Yes, but the concrete class can only extend an abstract class if it is in the same package.

Answer: a) Yes, but the concrete class must override all the abstract methods from the abstract class.

10. Consider the following code:

```java
abstract class Bird {
    abstract void fly();

    void makeSound() {
        System.out.println("Chirp Chirp");
    }
}

class Sparrow extends Bird {
    void fly() {
        System.out.println("Sparrow flies.");
    }
}

class Penguin extends Bird {
    void fly() {
        System.out.println("Penguins cannot fly.");
    }
    void makeSound() {
        System.out.println("Penguin sound.");
    }
}
```

What will be the output when the Main class is executed?

```java
public class Main {
    public static void main(String[] args) {
        Bird sparrow = new Sparrow();
        Bird penguin = new Penguin();
        sparrow.fly();
        penguin.fly();
        sparrow.makeSound();
        penguin.makeSound();
    }
}
```

a)
```
Sparrow flies.
Penguins cannot fly.
Chirp Chirp
```

Penguin sound.
```

b)
```
Sparrow flies.
Penguins cannot fly.
Penguin sound.
Penguin sound.
```

c)
```
Sparrow flies.
Penguin sound.
Chirp Chirp
Penguin sound.
```

d)
```
Sparrow flies.
Penguin sound.
Chirp Chirp
Penguins cannot fly.
```

Answer: a)
```
Sparrow flies.
Penguins cannot fly.
Chirp Chirp
Penguin sound.
```

**7.2 Interface (Implementing Multiple Interfaces)**

1. What is an interface in Java?
   a) A class that cannot be instantiated and serves as a blueprint for other classes.
   b) A type of data structure

 used to store collections of elements.
   c) A collection of abstract methods and constants without method implementations.
   d) A built-in data type used to store floating-point numbers.

Answer: c) A collection of abstract methods and constants without method implementations.

2. Which keyword is used to define an interface in Java?
   a) interface
   b) abstract
   c) class
   d) implements

Answer: a) interface

3. Can an interface contain concrete (non-abstract) methods in Java?
   a) Yes, an interface can only contain concrete methods.
   b) No, an interface can only contain abstract methods.
   c) Yes, an interface can have both abstract and concrete methods.
   d) No, an interface cannot have any methods.
```

Answer: b) No, an interface can only contain abstract methods.

4. In Java, can a class extend multiple interfaces?
   a) Yes, a class can extend multiple interfaces using the "extends" keyword.
   b) Yes, but only if the interfaces do not have any abstract methods.
   c) No, a class can only extend a single interface.
   d) No, a class cannot extend an interface.

Answer: a) Yes, a class can extend multiple interfaces using the "extends" keyword.

5. What is the main purpose of using interfaces in Java?
   a) To enforce encapsulation of data members in a class.
   b) To allow multiple inheritance from different classes.
   c) To provide a default implementation for methods in a class.
   d) To define a contract for classes that implement the interface.

Answer: d) To define a contract for classes that implement the interface.

6. Can an interface have variables with different access modifiers (e.g., public, private, protected) in Java?
   a) Yes, but only if the variables are declared as final.
   b) Yes, there are no restrictions on the access modifiers for variables in an interface.
   c) No, all variables in an interface must have the same access modifier.
   d) No, an interface cannot have any variables.

Answer: b) Yes, there are no restrictions on the access modifiers for variables in an interface.

7. Can a class be both an abstract class and implement an interface in Java?
   a) Yes, but only if the interface has at least one abstract method.
   b) Yes, and the abstract class must implement all the abstract methods from the interface.
   c) No, an abstract class cannot implement an interface.
   d) Yes, but an abstract class can only implement interfaces with concrete methods.

Answer: b) Yes, and the abstract class must implement all the abstract methods from the interface.

8. In Java, can an interface extend another interface?
   a) Yes, and the subclass must implement all the abstract methods from both interfaces.
   b) Yes, and the subclass can choose which abstract methods to implement from each interface.
   c) No, Java does not allow an interface to extend another interface.
   d) No, an interface can only extend a concrete class.

Answer: a) Yes, and the subclass must implement all the abstract methods from both interfaces.

9. What happens if a class implements an interface but fails to implement all the abstract methods defined in that interface?
   a) The class will be considered an abstract class and cannot be instantiated.
   b) The class will automatically inherit the implementation of the missing methods from the Object class.
   c) The code will not compile, resulting in a compilation error.
   d) The abstract methods will be implemented using default values defined in the interface.

Answer: c) The code will not compile, resulting in a compilation error.

10. Which of the following statements about interfaces is true?
   a) An interface can be directly instantiated using the "new" keyword.
   b) An interface must have at least one abstract method.
   c) An interface can only have static methods and variables.
   d) An interface can have non-static (instance) methods.

Answer: d) An interface can have non-static (instance) methods.

11. In Java, can an interface extend multiple interfaces?
    a) Yes, Java supports multiple inheritance for interfaces.
    b) Yes, but the interface must implement all the abstract methods from each extended interface.
    c) No, Java does not support multiple inheritance for interfaces.
    d) Yes, but an interface can only extend two other interfaces at most.

Answer: a) Yes, Java supports multiple inheritance for interfaces.

12. What is the purpose of declaring a method in an interface as "default" in Java?
    a) To specify that the method must be implemented in all classes that implement the interface.
    b) To provide a default implementation for the method in case a class does not override it.
    c) To prevent any class from implementing the interface.
    d) To make the method accessible only within the package.

Answer: b) To provide a default implementation for the method in case a class does not override it.

13. In Java, what is the purpose of using the "implements" keyword in a class declaration?
    a) To define an abstract class.
    b) To extend a superclass.
    c) To specify that the class is implementing one or more interfaces.
    d) To override a method from the superclass.

Answer: c) To specify that the class is implementing one or more interfaces.

14. Can an interface have static methods in Java?
    a) Yes, but only if the interface is declared as abstract.
    b) Yes, and all methods in an interface are implicitly static.
    c) No, an interface cannot have any static methods.
    d) Yes, but only if the interface is declared as final.

Answer: c) No, an interface cannot have any static methods.

15. In Java, can an abstract class implement an interface?
    a) Yes, but only if the interface has at least one abstract method.
    b) Yes, and the abstract class must implement all the abstract methods from the interface.
    c) No, an abstract class cannot implement an interface.
    d) Yes, but an abstract class can only implement interfaces with concrete methods.

Answer: b) Yes, and the abstract class must implement all the abstract methods from the interface.

8. Lecture: Final Variables, Final Methods, and Final Class

8.1 Final Variables, Final Methods, and Final Class

1. MCQ: Which keyword is used in Java to declare a final variable?
    a) `var`
    b) `final`
    c) `const`
    d) `static`

    Answer: b) `final`

2. MCQ: What is the purpose of a final method in Java?
    a) It cannot be overridden by subclasses.
    b) It cannot be called from other classes.

c) It can be overridden by subclasses but cannot be called.
d) It can be called, but its return type must be final as well.

Answer: a) It cannot be overridden by subclasses.

3. MCQ: Can a final class be extended (inherited) in Java?
a) Yes, but only if it has no methods.
b) Yes, but only if it has no variables.
c) No, a final class cannot be extended.
d) Yes, a final class can be extended normally like any other class.

Answer: c) No, a final class cannot be extended.

4. MCQ: What is the primary benefit of using final variables?
a) They consume less memory.
b) They cannot be modified once assigned.
c) They can be accessed from any method without restriction.
d) They can be used without initializing them.

Answer: b) They cannot be modified once assigned.

5. MCQ: Which of the following is true about the initialization of final variables?
a) Final variables must be initialized at the time of declaration.
b) Final variables can only be initialized using a constructor.
c) Final variables can be initialized anywhere in the code.
d) Final variables are automatically initialized to their default values.

Answer: a) Final variables must be initialized at the time of declaration.

6. MCQ: Can a method be both static and final in Java?
a) Yes, but it must have a void return type.
b) Yes, but it can only be called from the main method.
c) No, a method cannot be both static and final simultaneously.
d) Yes, a method can be both static and final.

Answer: d) Yes, a method can be both static and final.

7. MCQ: Which of the following modifiers can be used together with the final keyword?
a) synchronized
b) abstract
c) private
d) transient

Answer: c) private

8.2 Functional Interface

1. MCQ: How many abstract methods are allowed in a functional interface?
a) None
b) One
c) Two
d) As many as needed

Answer: b) One

2. MCQ: Which annotation is used to indicate that an interface is a functional interface?
a) @Functional
b) @FunctionalInterface
c) @FunctionalOnly
d) @SingleAbstractMethod

Answer: b) @FunctionalInterface

3. MCQ: What is the main purpose of functional interfaces in Java?
a) To represent classes that are functional and have only static methods.
b) To indicate that the interface can only have abstract methods.
c) To enable functional programming features like lambda expressions and method references.
d) To restrict the interface from being extended by other interfaces.

Answer: c) To enable functional programming features like lambda expressions and method references.

4. MCQ: Which of the following is a valid example of a functional interface?
a) `public interface MyInterface { void method1(); void method2(); }`
b) `public interface MyInterface { void method1() { } }`
c) `public interface MyInterface { int method1(int a, int b); }`
d) `public interface MyInterface { default void method1() { } }`

Answer: c) `public interface MyInterface { int method1(int a, int b); }`

5. MCQ: Can a functional interface have more than one default method?
a) Yes, as many as needed.
b) No, it can have only one default method.
c) Yes, but only if they are marked with the `@Default` annotation.
d) No, functional interfaces cannot have default methods.

Answer: b) No, it can have only one default method.

6. MCQ: How does the lambda expression in Java relate to functional interfaces?
a) Lambda expressions can only be used with functional interfaces.
b) Lambda expressions are used to define abstract classes in Java.
c) Lambda expressions are a replacement for constructors in functional interfaces.
d) Lambda expressions can be used with any Java class or interface.

Answer: a) Lambda expressions can only be used with functional interfaces.

7. MCQ: Which functional interface is provided by Java to represent a supplier of values?
a) `Consumer<T>`
b) `Predicate<T>`
c) `Supplier<T>`
d) `Function<T, R>`

Answer: c) `Supplier<T>`

8.3 New Interface Features (Java 8 & Above)

1. MCQ: What is the purpose of default methods in interfaces?
a) To allow interfaces to be instantiated.
b) To provide a default implementation for a method that can be overridden.
c) To mark a method as final and prevent overriding.
d) To create static methods within interfaces.

Answer: b) To provide a default implementation for a method that can be overridden.

2. MCQ: Can a class implement multiple interfaces that have default methods with the same signature?
a) Yes, but it will cause a compilation error.
b) No, a class cannot implement interfaces with default methods.
c) Yes, but the class must provide an implementation for the conflicting methods.
d) Yes, but only if the interfaces are in the same package.

Answer: c) Yes, but the class must provide an implementation for the conflicting methods.

3. MCQ: What is the keyword used in interfaces to allow methods to be defined only by concrete (non-abstract) subclasses?
a) `subclass`
b) `extends`
c) `sealed`
d) `default`

Answer: c) `sealed`

4. MCQ: Which of the following is true about static methods in interfaces?
a) Static methods in interfaces are inherited by implementing classes.
b) Static methods in interfaces can be overridden by implementing classes.
c) Static methods in interfaces can be called directly using the interface name.
d) Static methods in interfaces are automatically synchronized.

Answer: c) Static methods in interfaces can be called directly using the interface name.

5. MCQ: What is the purpose of private methods in interfaces?
a) They are used to prevent other interfaces from accessing the methods.
b) They are used to define helper methods that can be reused in default methods.
c) Private methods in interfaces are not allowed; only public methods are allowed.
d) Private methods in interfaces are equivalent to final methods.

Answer: b) They are used to define helper methods that can be reused in default methods.

6. MCQ: How are

static methods in interfaces different from regular static methods in classes?
a) Static methods in interfaces can only be accessed within the same package.
b) Static methods in interfaces cannot be called directly; they can only be inherited.
c) Static methods in interfaces are inherited by implementing classes, while regular static methods are not.
d) Static methods in interfaces cannot have return values.

Answer: c) Static methods in interfaces are inherited by implementing classes, while regular static methods are not.

7. MCQ: Which Java version introduced the concept of default methods in interfaces?
a) Java 5
b) Java 6
c) Java 7
d) Java 8

Answer: d) Java 8

8.4 Arrays

1. MCQ: In Java, what is the index of the first element in an array?
a) -1
b) 0
c) 1
d) The index is not specified; it depends on the array size.

Answer: b) 0

2. MCQ: How do you access the length of an array in Java?
a) `array.size()`
b) `array.length`

c) `array.size`
d) `array.length()`

Answer: b) `array.length`

3. MCQ: What happens when you try to access an element outside the bounds of an array?
a) The program crashes with an "ArrayIndexOutOfBoundsException."
b) The program continues execution, and the element is initialized to 0.
c) The program automatically expands the array to include the new element.
d) The program displays an error message but continues execution.

Answer: a) The program crashes with an "ArrayIndexOutOfBoundsException."

4. MCQ: How do you create a two-dimensional array in Java?
a) `int[] array = new int[rows, columns];`
b) `int array = new int[rows][columns];`
c) `int[][] array = new int[rows][columns];`
d) `int array = new int[rows, columns];`

Answer: c) `int[][] array = new int[rows][columns];`

5. MCQ: What is the default value of an uninitialized element in a numeric array?
a) 0
b) 1
c) -1
d) null

Answer: a) 0

6. MCQ: Can an array in Java store elements of different data types?
a) Yes, but only if the array is explicitly declared as "mixed."
b) No, all elements in an array must be of the same data type.
c) Yes, but only if the array is declared as a "variant" array.
d) Yes, but only if the elements are explicitly cast to a common base type.

Answer: b) No, all elements in an array must be of the same data type.

7. MCQ: What is the highest index of an array with 10 elements?
a) 9
b) 10
c) 11
d) It depends on the data type of the elements.

Answer: a) 9

8.5 Enumerations

1. MCQ: In Java, what keyword is used to declare an enumeration?
a) `enum`
b) `enumType`
c) `enumeration`
d) `type`

Answer: a) `enum`

2. MCQ: Which of the following is a valid declaration of an enumeration in Java?
a) `enum Days = {Monday, Tuesday, Wednesday};`
b) `enum Days = (Monday, Tuesday, Wednesday);`
c) `enum Days { Monday, Tuesday, Wednesday }`
d) `enum Days: Monday, Tuesday, Wednesday;`

Answer: c) `enum Days { Monday, Tuesday, Wednesday }`

3. MCQ: Can an enumeration in Java have methods?
   a) Yes, but only abstract methods.
   b) Yes, but only static methods.
   c) No, enumerations cannot have methods.
   d) Yes, both abstract and static methods.

   Answer: d) Yes, both abstract and static methods.

4. MCQ: How do you access an enumeration constant in Java?
   a) By using the `.` operator, e.g., `Days.Monday`
   b) By using the `->` operator, e.g., `Days->Monday`
   c) By using the `::` operator, e.g., `Days::Monday`
   d) By using the `=>` operator, e.g., `Days=>Monday`

   Answer: a) By using the `.` operator, e.g., `Days.Monday`

5. MCQ: Can an enumeration have constructors in Java?
   a) No, enumerations cannot have constructors.
   b) Yes, but the constructors must be private.
   c) Yes, and they must be public.
   d) Yes, and they must be static.

   Answer: b) Yes, but the constructors must be private.

6. MCQ: What is the data type of an enumeration constant in Java?
   a) `String`
   b) `int`
   c) `enum`
   d) The data type depends on the implementation.

   Answer: c) `enum`

7. MCQ: How do you iterate over all the constants of an enumeration in Java?
   a) Using a regular `for` loop
   b) Using a `foreach` loop
   c) Using the `enumValues()` method
   d) Using the `EnumSet` class

   Answer: c) Using the `enumValues()` method

8.6 Create an Array of Employee Class and Initialize Array Elements with Different Employee Objects. Try to Understand the Number of Objects on Heap Memory When Any Array is Created.

1. MCQ: When an array of `Employee` class is created, how many `Employee` objects are there in memory initially?
   a) One, and all array elements point to the same object.
   b) The number of objects depends on the size of the array.
   c) One object for each element of the array.
   d) Two objects: one for the array and one for the first element.

   Answer: b) The number of objects depends on the size of the array.

2. MCQ: How can you access the elements of the `Employee` array named `employees` in Java?
   a) `employees(index)`
   b) `employees(index).getValue()`
   c) `employees[index]`
   d) `employees.getValue(index)`

Answer: c) `employees[index]`

3. MCQ: If an `Employee` array is declared but not initialized, how many objects are on the heap memory?
a) None, objects are created on the heap only when the array is initialized.
b) One object for each potential element in the array, but all are `null`.
c) One object for the array, and the elements point to separate objects.
d) It depends on the default constructor of the `Employee` class.

Answer: a) None, objects are created on the heap only when the

array is initialized.

4. MCQ: How can you find the number of elements in the `employees` array in Java?
a) `employees.length()`
b) `employees.count()`
c) `employees.size()`
d) `employees.length`

Answer: d) `employees.length`

5. MCQ: What happens if you try to access an element of the `employees` array that is out of bounds?
a) The program will crash with an error message.
b) Java automatically extends the size of the array and initializes the element to `null`.
c) The program will display a warning but continue execution.
d) It depends on the compiler used.

Answer: a) The program will crash with an error message.

6. MCQ: How can you initialize an `Employee` array with values in Java?
a) `Employee[] employees = new Employee {new Employee("John"), new Employee("Alice")};`
b) `Employee[] employees = new Employee[] {"John", "Alice"};`
c) `Employee[] employees = {"John", "Alice"};`
d) `Employee[] employees = new Employee[] {new Employee("John"), new Employee("Alice")};`

Answer: d) `Employee[] employees = new Employee[] {new Employee("John"), new Employee("Alice")};`

7. MCQ: What is the default value of an uninitialized element in an `Employee` array?
a) `null`
b) `0`
c) An empty `Employee` object with default values for its fields.
d) It depends on the default constructor of the `Employee` class.

Answer: a) `null`

9. Lecture: Access Modifiers (public, private, protected, and default)

9.1 Access Modifiers (public, private, protected, and default)

1. MCQ: Which access modifier provides the widest accessibility in Java?
a) `public`
b) `private`
c) `protected`
d) default (no modifier)

Answer: a) `public`

2. MCQ: Which access modifier restricts access to within the same class only?
a) `public`
b) `private`
c) `protected`
d) default (no modifier)

Answer: b) `private`

3. MCQ: What is the default access modifier for class members (variables and methods) if no modifier is specified?
a) `public`
b) `private`
c) `protected`
d) default (package-private)

Answer: d) default (package-private)

4. MCQ: Can a `protected` member of a superclass be accessed from a subclass in a different package?
a) Yes, always.
b) No, `protected` members are accessible only within the same package.
c) Yes, but only if the subclass is in the same package as the superclass.
d) Yes, but only if the subclass is in a subclassing relationship with the superclass.

Answer: c) Yes, but only if the subclass is in the same package as the superclass.

5. MCQ: Which access modifier is often used for methods that should be accessible from anywhere in the code?
a) `public`
b) `private`
c) `protected`
d) default (no modifier)

Answer: a) `public`

6. MCQ: In Java, what is the access level of a class member if it is declared with no access modifier?
a) `public`
b) `private`
c) `protected`
d) default (package-private)

Answer: d) default (package-private)

7. MCQ: Can a class member with `private` access modifier be accessed from another class?
a) Yes, it can be accessed from any class in the same package.
b) Yes, but only if the classes are in a subclassing relationship.
c) No, `private` members are accessible only within the same class.
d) Yes, but only if the member is static.

Answer: c) No, `private` members are accessible only within the same class.

9.2 Packages and Import Statements

1. MCQ: What is the main purpose of using packages in Java?
a) To minimize the number of lines of code in a project.
b) To prevent access to certain classes and methods.
c) To organize classes and avoid naming conflicts.
d) To improve the performance of the code.

Answer: c) To organize classes and avoid naming conflicts.

2. MCQ: Which keyword is used to define a package in Java?
a) `package`
b) `import`
c) `namespace`
d) `module`

Answer: a) `package`

3. MCQ: What is the convention for naming packages in Java to avoid conflicts with packages from other developers?
a) Use all uppercase letters, e.g., `PACKAGE_NAME`.
b) Use a prefix based on the project name or the company domain, e.g., `com.company.project`.
c) Use a single letter, e.g., `p`.
d) Use a random combination of letters and numbers.

Answer: b) Use a prefix based on the project name or the company domain, e.g., `com.company.project`.

4. MCQ: How do you import a specific class from a package in Java?
a) `import package.ClassName;`
b) `import package;`
c) `import package.*;`
d) `import ClassName from package;`

Answer: a) `import package.ClassName;`

5. MCQ: If no package is specified at the beginning of a Java file, which package does it belong to?
a) `java.lang`
b) `java.util`
c) The file won't compile without a package declaration.
d) The default package (no package).

Answer: d) The default package (no package).

6. MCQ: Can two classes with the same name exist in different packages in Java?
a) Yes, but only if both classes are in the same project.
b) Yes, but the classes must have the same methods and fields.
c) No, two classes with the same name cannot exist in different packages.
d) Yes, but the classes must be in the same directory.

Answer: a) Yes, but only if both classes are in the same project.

7. MCQ: What happens if you try to use a class from another package without importing it in Java?
a) The code will not compile.
b) The code will compile, but the class will not be accessible.
c) The code will compile, and the class will be automatically imported.
d) The code will compile, but the class will be treated as if it's in the same package.

Answer: a) The code will not compile.

9.3 Static Imports

1. MCQ: What is the purpose of static imports in Java?
a) To import classes from other packages as static members.
b) To import all classes and methods from a package as static.
c) To access static members of a class without specifying the class name.
d) To enable importing classes from non-static contexts.

Answer: c) To access static members of a class without specifying the class name.

2. MCQ: Which keyword is used for static imports in Java?
a) `import`
b) `use`
c) `import static`
d) `static`

Answer: c) `import static`

3. MCQ: When should you use static imports in Java?
a) Whenever possible, to reduce code verbosity and improve readability.
b) Only when you want to access non-static members of a class.
c) Only when the static members have the same names as members in the current class.
d) Only when importing multiple classes from the same package.

Answer: a) Whenever possible, to reduce code verbosity and improve readability.

4. MCQ: Which of the following is a valid syntax for a static import statement in Java?
a) `import static java.util.*;`
b) `static import java.util.*;`
c) `import static java.util.ArrayList.*;`
d) `static import java.util.ArrayList.*;`

Answer: d) `static import java.util.ArrayList.*;`

5. MCQ: What happens if there is a naming conflict between a static member and an instance member in Java?
a) The instance member will take precedence, and the static member will be inaccessible.
b) The static member will take precedence, and the instance member will be inaccessible.
c) The code will not compile until the conflict is resolved.
d) Both members can be accessed using their respective class names.

Answer: a) The instance member will take precedence, and the static

member will be inaccessible.

6. MCQ: How are static imports different from regular imports in Java?
a) Static imports are used only for importing static members, while regular imports are used for non-static members.
b) Static imports are used to access members without specifying the class name, while regular imports are used to avoid naming conflicts.
c) Static imports are required for all classes used in the code, while regular imports are optional.
d) Static imports are used to import classes from other packages, while regular imports are used to import classes from the same package.

Answer: b) Static imports are used to access members without specifying the class name, while regular imports are used to avoid naming conflicts.

7. MCQ: Can you use static imports to import non-static members in Java?
a) Yes, but only if the non-static members are marked as `final`.
b) Yes, but only if the non-static members are marked as `protected`.
c) No, static imports are limited to importing static members only.
d) Yes, but only if the non-static members are declared in the same package as the importing class.

Answer: c) No, static imports are limited to importing static members only.

9.4 Constructor Chaining (With and Without Packages)

1. MCQ: What is constructor chaining in Java?
   a) The process of creating multiple constructors for a class.
   b) The process of calling one constructor from another within the same class or from a superclass.
   c) The process of using static constructors to initialize static variables.
   d) The process of using interfaces to enforce constructor implementation.

   Answer: b) The process of calling one constructor from another within the same class or from a superclass.

2. MCQ: In constructor chaining, which keyword is used to call another constructor from within a constructor?
   a) `super`
   b) `this`
   c) `new`
   d) `chaining`

   Answer: b) `this`

3. MCQ: How do you call a superclass constructor from a subclass constructor in Java?
   a) `call super();`
   b) `super();`
   c) `invoke super();`
   d) `superclass();`

   Answer: b) `super();`

4. MCQ: When is it necessary to use constructor chaining in Java?
   a) Constructor chaining is always necessary for all classes.
   b) Constructor chaining is necessary when there are multiple constructors in a class.
   c) Constructor chaining is necessary when the class is in a different package.
   d) Constructor chaining is necessary when the class has no instance variables.

   Answer: b) Constructor chaining is necessary when there are multiple constructors in a class.

5. MCQ: What is the role of the `this` keyword in constructor chaining?
   a) It allows access to the superclass constructor.
   b) It allows access to the current class constructor.
   c) It helps avoid naming conflicts between instance variables and parameters.
   d) It is used to define static constructors.

   Answer: c) It helps avoid naming conflicts between instance variables and parameters.

6. MCQ: In constructor chaining, which constructor is called first?
   a) The default constructor (if available).
   b) The constructor with the most parameters.
   c) The constructor with no parameters.
   d) The constructor with the fewest parameters.

   Answer: c) The constructor with no parameters.

7. MCQ: Can constructor chaining be used to call a private constructor in the same class?
   a) Yes, but only if the constructors have the same number of parameters.
   b) Yes, constructor chaining can always access private constructors.
   c) No, constructor chaining cannot access private constructors.
   d) Yes, but only if the constructors have different access modifiers.

   Answer: b) Yes, constructor chaining can always access private constructors.

9.5 Accessing Protected Variables and Methods Outside the Package

1. MCQ: What is the accessibility of `protected` variables and methods in Java?
   a) They are accessible only within the same class.
   b) They are accessible within the same package and by subclasses in any package.
   c) They are accessible within the same package only.
   d) They are accessible by any class in any package.

   Answer: b) They are accessible within the same package and by subclasses in any package.

2. MCQ: Can a `protected` method in a superclass be accessed from a subclass in a different package?
   a) Yes, always.
   b) No, `protected` methods can only be accessed within the same package.
   c) Yes, but only if the subclass is in the same package as the superclass.
   d) Yes, but only if the subclass is in a subclassing relationship with the superclass.

   Answer: c) Yes, but only if the subclass is in the same package as the superclass.

3. MCQ: How is the accessibility of `protected` variables different from `protected` methods in Java?
   a) `protected` variables are accessible from any package, while `protected` methods are not.
   b) `protected` methods are accessible from any package, while `protected` variables are not.
   c) There is no difference; both `protected` variables and methods are accessible from any package.
   d) `protected` variables and methods have the same accessibility within the same package but are inaccessible outside the package.

   Answer: b) `protected` methods are accessible from any package, while `protected` variables are not.

4. MCQ: Can a non-subclass, non-package class access a `protected` member of another class?
   a) Yes, but only if the member is declared as `final`.
   b) Yes, but only if the member is declared as `static`.
   c) Yes, but only if the member is explicitly cast to the correct type.
   d) No, `protected` members are accessible only by subclasses and classes in the same package.

   Answer: d) No, `protected` members are accessible only by subclasses and classes in the same package.

5. MCQ: How do you access a `protected` member from a subclass in a different package?
   a) By importing the superclass package and using the `protected` member directly.
   b) By using the `super` keyword followed by the `protected` member's name.
   c) By casting the subclass object to the superclass type and accessing the `protected` member.
   d) By using the `protected` member directly without any additional steps.

   Answer: b) By using the `super` keyword followed by the `protected` member's name.

6. MCQ: What happens if you try to access a `protected` member from a non-subclass and non-package class?
   a) The code will not compile.
   b) The code will compile, but the `protected` member will be inaccessible.
   c) The code will compile, but the `protected` member will be `null`.
   d) The code will compile, but the `protected` member will have default (package-private) access.

   Answer: a) The code will not compile.

7. MCQ: How can you access a `protected` member of a class if you are not in a subclass or the same package?
   a) Use reflection to access the member dynamically.
   b) Declare a method in the class that returns the `protected` member.
   c) Change the access modifier of the member to `public`.
   d) Move the accessing class to the same package as the class containing the `protected` member.

   Answer: d) Move the accessing class to the same package as the class containing the `protected` member.

10. Lecture: Garbage Collection in Java

10.1 Requesting JVM to Run Garbage Collection

1. MCQ: Which method can be used to request the JVM to run garbage collection in Java?
   a) `System.gc()`
   b) `Runtime.gc()`
   c) `JVM.gc()`
   d) `GarbageCollector.run()`

   Answer: a) `System.gc()`

2. MCQ: What is the purpose of requesting garbage collection in Java?
   a) To force the JVM to immediately free up memory.
   b) To clean up unused objects and reclaim memory.
   c) To automatically optimize memory usage in the application.
   d) To stop the execution of the Java program.

   Answer: b) To clean up unused objects and reclaim memory.

3. MCQ: Is it guaranteed that the JVM will run garbage collection immediately after `System.gc()` is called?
   a) Yes, the JVM will run garbage collection immediately.
   b) No, the JVM may choose to run garbage collection at its own discretion.
   c) Yes, but only if there are no active threads in the application.
   d) No, the `System.gc()` method does not trigger garbage collection.

   Answer: b) No, the JVM may choose to run garbage collection at its own discretion.

4. MCQ: What are the potential downsides of explicitly calling `System.gc()`?
   a) It may lead to a memory leak.
   b) It may slow down the application's performance.
   c) It may cause a runtime exception.
   d) There are no downsides to calling `System.gc()`.

   Answer: b) It may slow down the application's performance.

5. MCQ (Coding): Which data type is used to represent a reference variable that can be garbage collected?
   a) `int`
   b) `String`
   c) `double`
   d) `Object`

   Answer: d) `Object`

6. MCQ: When is it appropriate to call `System.gc()` in Java?
   a) Whenever the application is running out of memory.
   b) When there are unused objects that need to be freed up immediately.
   c) As part of regular application maintenance tasks.
   d) It is not recommended to manually call `System.gc()`.

   Answer: d) It is not recommended to manually call `System.gc()`.

7. MCQ: What does the JVM do during garbage collection?
   a) It deallocates memory for all objects in the application.
   b) It identifies and collects unused objects to free up memory.
   c) It optimizes the performance of the application.
   d) It stops all running threads to clean up the memory.

Answer: b) It identifies and collects unused objects to free up memory.

10.2 Different Ways to Make Object Eligible for Garbage Collection: (Nulling a Reference Variable, Re-assigning a Reference Variable & Island of Isolation)

1. MCQ: How can you make an object eligible for garbage collection in Java?
   a) By setting its reference variable to `null`.
   b) By calling the `System.gc()` method.
   c) By creating a new reference variable for the same object.
   d) By reassigning its reference variable to another object.

   Answer: a) By setting its reference variable to `null`.

2. MCQ (Coding): Which of the following code snippets correctly makes an object eligible for garbage collection?

```java
// Option 1
Object obj = new Object();
obj = null;

// Option 2
Object obj = new Object();
Object obj2 = obj;
obj = null;

// Option 3
Object obj = new Object();
obj = new Object();
```

   a) Option 1
   b) Option 2
   c) Option 3
   d) None of the above

   Answer: b) Option 2

3. MCQ: What is the purpose of the "Island of Isolation" in the context of garbage collection?
   a) It refers to a group of objects with circular references that cannot be garbage collected.
   b) It is a term used to describe objects that are not eligible for garbage collection.
   c) It is a garbage collection algorithm used by the JVM.
   d) It refers to a group of objects with strong references that can be garbage collected together.

   Answer: a) It refers to a group of objects with circular references that cannot be garbage collected.

4. MCQ: When does an object become eligible for garbage collection if all of its references are set to `null`?
   a) Immediately after the last reference is set to `null`.
   b) The object becomes eligible for garbage collection as soon as it is not reachable by any live thread.
   c) The object will never be eligible for garbage collection.
   d) It depends on the garbage collector's scheduling algorithm.

   Answer: b) The object becomes eligible for garbage collection as soon as it is not reachable by any live thread.

5. MCQ: How does nulling a reference variable help in garbage collection?
   a) It immediately deallocates the memory occupied by the object.
   b) It marks the object as garbage, and

the JVM will collect it during the next garbage collection cycle.
c) Nulling a reference variable has no effect on garbage collection.
d) It prevents the object from being garbage collected.

Answer: b) It marks the object as garbage, and the JVM will collect it during the next garbage collection cycle.

6. MCQ (Coding): What is the output of the following code snippet?

```java
public class GarbageCollectionDemo {
    public static void main(String[] args) {
        Object obj1 = new Object();
        Object obj2 = new Object();
        obj1 = obj2;
        obj2 = null;
        System.gc();
    }
}
```

a) The code will not compile because of the call to `System.gc()`.
b) The code will compile, but it will throw a runtime exception.
c) The code will compile, and the objects will be eligible for garbage collection.
d) The code will compile, but it will have no effect on garbage collection.

Answer: c) The code will compile, and the objects will be eligible for garbage collection.

7. MCQ: What happens if an object is part of an "Island of Isolation" in Java?
a) The object is immediately garbage collected.
b) The object remains in memory until the application terminates.
c) The JVM throws a runtime exception.
d) The object becomes eligible for garbage collection when the island is identified.

Answer: b) The object remains in memory until the application terminates.

10.3 Finalize Method

1. MCQ: What is the purpose of the `finalize()` method in Java?
a) It is used to make an object eligible for garbage collection.
b) It is used to clean up resources before an object is garbage collected.
c) It is a method that is automatically called by the JVM during garbage collection.
d) It is a reserved keyword and cannot be used in Java code.

Answer: b) It is used to clean up resources before an object is garbage collected.

2. MCQ (Coding): Which of the following code snippets correctly demonstrates the use of the `finalize()` method?

```java
// Option 1
public class MyClass {
    // ... class code ...

    public void finalize() {
        // clean up resources here
    }
}
```

```
// Option 2
public class MyClass {
    // ... class code ...

    public void finalize(Object obj) {
        // clean up resources here
    }
}

// Option 3
public class MyClass {
    // ... class code ...

    public void finalizing() {
        // clean up resources here
    }
}
```

a) Option 1
b) Option 2
c) Option 3
d) None of the above

Answer: a) Option 1

3. MCQ: When does the `finalize()` method get called in Java?
a) The `finalize()` method is automatically called when an object is created.
b) The `finalize()` method is automatically called when an object is garbage collected.
c) The `finalize()` method must be called manually by the programmer.
d) The `finalize()` method is automatically called when an object is set to `null`.

Answer: b) The `finalize()` method is automatically called when an object is garbage collected.

4. MCQ: What is the return type of the `finalize()` method in Java?
a) `void`
b) `boolean`
c) `int`
d) `Object`

Answer: a) `void`

5. MCQ: What happens if an uncaught exception is thrown inside the `finalize()` method?
a) The JVM will automatically handle the exception and continue garbage collection.
b) The exception will be propagated to the calling code that triggered garbage collection.
c) The JVM will skip the `finalize()` method and proceed with garbage collection.
d) The JVM will terminate the application.

Answer: c) The JVM will skip the `finalize()` method and proceed with garbage collection.

6. MCQ: Is it recommended to use the `finalize()` method for resource cleanup in Java?
a) Yes, it is a best practice to use `finalize()` for resource cleanup.
b) No, `finalize()` is deprecated, and other approaches should be used for resource cleanup.
c) It depends on the specific use case and requirements of the application.
d) Yes, `finalize()` is the only method available for resource cleanup in Java.

Answer: b) No, `finalize()` is deprecated, and other approaches should be used for resource cleanup.

7. MCQ: What can be done inside the `finalize()` method to ensure proper resource cleanup?
a) Call the `System.gc()` method to trigger garbage collection.

b) Close open files, release network connections, or free up other resources.
c) Set the object's reference to `null`.
d) Trigger a runtime exception to indicate cleanup completion.

Answer: b) Close open files, release network connections, or free up other resources.

10.4 Create a Demo Application to Understand the Role of Access Modifiers.

1. MCQ (Coding): Given the following Java classes, what will be the output of the demo application?

```java
// File: MyClass.java
public class MyClass {
    public int publicVar = 10;
    private int privateVar = 20;
    protected int protectedVar = 30;
    int defaultVar = 40;

    public void display() {
        System.out.println("Public: " + publicVar);
        System.out.println("Private: " + privateVar);
        System.out.println("Protected: " + protectedVar);
        System.out.println("Default: " + defaultVar);
    }
}

// File: Main.java
public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.display();
    }
}
```

a) The code will not compile due to access modifier conflicts.
b) The code will compile, and all variables will be accessible and displayed.
c) The code will compile, but only public variables will be accessible and displayed.
d) The code will compile, but only protected and public variables will be accessible and displayed.

Answer: c) The code will compile, but only public variables will be accessible and displayed.

2. MCQ: What is the output of the demo application if `Main.java` is in a different package than `MyClass.java`?

a) The code will not compile because `MyClass` is not accessible.
b) The code will compile, but only public variables will be accessible and displayed.
c) The code will compile, but all variables will be accessible and displayed.
d) The code will compile, but only protected and public variables will be accessible and displayed.

Answer: b) The code will compile, but only public variables will be accessible and displayed.

3. MCQ: Which access modifier(s) allow a class member to be accessible from any class in any package?
a) `public`
b) `protected`
c) `private`
d) default (package-private)

Answer: a) `public`

4. MCQ (Coding): What modification is needed in `MyClass.java

` to make the `protectedVar` accessible from the `Main` class, which is in a different package?

a) Change `protected` to `public` for `protectedVar`.
b) Add the `public` modifier before `class MyClass`.
c) Add the `protected` modifier before `class MyClass`.
d) There is no way to access `protectedVar` from a different package.

Answer: a) Change `protected` to `public` for `protectedVar`.

5. MCQ: If the `privateVar` in `MyClass.java` is changed to `protected`, what will be the output of the demo application?

a) The code will not compile because `privateVar` is not accessible.
b) The code will compile, but only public and protected variables will be accessible and displayed.
c) The code will compile, but only public variables will be accessible and displayed.
d) The code will compile, and all variables will be accessible and displayed.

Answer: b) The code will compile, but only public and protected variables will be accessible and displayed.

6. MCQ (Coding): What happens if `defaultVar` in `MyClass.java` is changed to `protected`?

a) The code will not compile due to access modifier conflicts.
b) The code will compile, but only public and protected variables will be accessible and displayed.
c) The code will compile, but only public variables will be accessible and displayed.
d) The code will compile, and all variables will be accessible and displayed.

Answer: a) The code will not compile due to access modifier conflicts.

7. MCQ: What is the main purpose of using access modifiers in Java?
a) To prevent access to certain classes and methods.
b) To minimize the number of lines of code in a project.
c) To organize classes and avoid naming conflicts.
d) To improve the performance of the code.

Answer: a) To prevent access to certain classes and methods.

10.5 Implement Multilevel Inheritance Using Different Packages.

1. MCQ (Coding): Given the following Java classes in different packages, what is the output of the demo application?

```java
// File: Package1/Grandparent.java
package Package1;

public class Grandparent {
    protected void display() {
        System.out.println("This is the Grandparent class.");
    }
}

// File: Package2/Parent.java
package Package2;
import Package1.Grandparent;

public class Parent extends Grandparent {
```

```
    protected void display() {
        System.out.println("This is the Parent class.");
    }
}

// File: Main.java
import Package2.Parent;

public class Main {
    public static void main(String[] args) {
        Parent obj = new Parent();
        obj.display();
    }
}
```

a) The code will not compile due to package-private access conflicts.
b) The code will compile, but the output will be "This is the Parent class."
c) The code will compile, but the output will be "This is the Grandparent class."
d) The code will compile, but there will be a runtime exception.

Answer: b) The code will compile, but the output will be "This is the Parent class."

2. MCQ: What happens if the `display()` method in `Grandparent.java` is changed from `protected` to `private`?

a) The code will not compile due to access modifier conflicts.
b) The code will compile, but there will be a runtime exception.
c) The code will compile, but the output will be "This is the Grandparent class."
d) The code will compile, but the output will be "This is the Parent class."

Answer: a) The code will not compile due to access modifier conflicts.

3. MCQ (Coding): What is the output of the demo application if `Parent.java` is modified as follows?

```java
// File: Package2/Parent.java
package Package2;
import Package1.Grandparent;

public class Parent extends Grandparent {
    public void display() {
        System.out.println("This is the Parent class.");
    }
}
```

a) The code will not compile due to package-private access conflicts.
b) The code will compile, but there will be a runtime exception.
c) The code will compile, but the output will be "This is the Grandparent class."
d) The code will compile, but the output will be "This is the Parent class."

Answer: d) The code will compile, but the output will be "This is the Parent class."

4. MCQ: What type of inheritance relationship is depicted in the code between `Grandparent`, `Parent`, and `Main` classes?
a) Single inheritance
b) Multiple inheritance
c) Multilevel inheritance
d) Hierarchical inheritance

Answer: c) Multilevel inheritance

5. MCQ (Coding): What is the output of the demo application if `Parent.java` is modified as follows?

```java
// File: Package2/Parent.java
package Package2;
import Package1.Grandparent;

public class Parent extends Grandparent {
    protected void display() {
        System.out.println("This is the Parent class.");
        super.display();
    }
}
```

a) The code will not compile due to package-private access conflicts.
b) The code will compile, but there will be a runtime exception.
c) The code will compile, but the output will be "This is the Grandparent class."
d) The code will compile, and the output will be "This is the Parent class. This is the Grandparent class."

Answer: d) The code will compile, and the output will be "This is the Parent class. This is the Grandparent class."

6. MCQ: What is the main benefit of multilevel inheritance in Java?
a) It allows a class to inherit from multiple superclasses.
b) It allows a class to have multiple subclasses.
c) It allows a class to access members of multiple classes.
d) It provides a hierarchical structure for class inheritance.

Answer: d) It provides a hierarchical structure for class inheritance.

7. MCQ: Can you implement multilevel inheritance with more than three classes in Java?
a) No, multilevel inheritance is limited to exactly three classes.
b) Yes, you can implement multilevel inheritance with any number of classes.
c) Yes, but it is limited to three levels of inheritance only.
d) Yes, but each subclass can have only one superclass.

Answer: b) Yes, you can implement multilevel inheritance with any number of classes.

10.6 Access/Invoke Protected Members/Methods of a Class Outside the Package.

1. MCQ (Coding): Given the following Java classes, what is the output of the demo application?

```java
// File: Package1/MyClass.java
package Package1;

public class MyClass {
    protected void display() {
        System.out.println("This is a protected method.");
    }
}

// File: Main.java
import Package1.MyClass;
```

```
public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.display

();
    }
}
```

a) The code will not compile because the `display()` method is not accessible.
b) The code will compile, and "This is a protected method." will be printed.
c) The code will compile, but there will be a runtime exception.
d) The code will compile, but the `display()` method will not be invoked.

Answer: a) The code will not compile because the `display()` method is not accessible.

2. MCQ: How can you access a protected method of a class outside the package in Java?
a) Use the `private` modifier instead of `protected`.
b) Use the `public` modifier instead of `protected`.
c) Move the accessing class to the same package as the class containing the protected method.
d) Use the `protected` modifier and extend the class containing the protected method.

Answer: d) Use the `protected` modifier and extend the class containing the protected method.

3. MCQ (Coding): What is the output of the demo application if `Main.java` is modified as follows?

```java
// File: Main.java
import Package1.MyClass;

public class Main extends MyClass {
    public static void main(String[] args) {
        Main obj = new Main();
        obj.display();
    }
}
```

a) The code will not compile due to access modifier conflicts.
b) The code will compile, but "This is a protected method." will not be printed.
c) The code will compile, and "This is a protected method." will be printed.
d) The code will compile, but there will be a runtime exception.

Answer: c) The code will compile, and "This is a protected method." will be printed.

4. MCQ: What is the main advantage of using the `protected` access modifier in Java?
a) It allows unrestricted access to the member in any package.
b) It restricts access to the member only within the same package.
c) It allows access to the member within the same package and by subclasses in any package.
d) It allows access to the member within the same package and by subclasses in the same package.

Answer: c) It allows access to the member within the same package and by subclasses in any package.

5. MCQ (Coding): What is the output of the demo application if `Main.java` is modified as follows?

```java
// File: Main.java
import Package1.MyClass;
```

```
public class Main {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        ((Main)obj).display();
    }
}
```

a) The code will not compile due to an invalid cast.
b) The code will compile, but there will be a runtime exception.
c) The code will compile, and "This is a protected method." will be printed.
d) The code will compile, but the `display()` method will not be invoked.

Answer: b) The code will compile, but there will be a runtime exception.

6. MCQ: Can protected members of a class be accessed using object references of the superclass?
a) Yes, it is possible to access protected members using the superclass reference.
b) No, protected members can only be accessed using subclass references.
c) Yes, but only if the superclass is in the same package as the subclass.
d) No, protected members cannot be accessed outside the package.

Answer: a) Yes, it is possible to access protected members using the superclass reference.

7. MCQ: What is the purpose of using the `protected` access modifier in Java?
a) To allow access to a member from any class in any package.
b) To restrict access to a member within the same package only.
c) To allow access to a member within the same package and by subclasses in any package.
d) To allow access to a member within the same package and by subclasses in the same package.

Answer: c) To allow access to a member within the same package and by subclasses in any package.

10.7 Override Finalize Method to Understand the Behavior of JVM Garbage Collector.

1. MCQ: Can you directly call the `finalize()` method in Java?
a) Yes, by using the `finalize()` method's name directly in the code.
b) Yes, by using the `this.finalize()` statement inside a method.
c) No, the `finalize()` method cannot be called directly; it is called automatically by the JVM.
d) No, the `finalize()` method is a private method and cannot be accessed.

Answer: c) No, the `finalize()` method cannot be called directly; it is called automatically by the JVM.

2. MCQ (Coding): Given the following Java class, what will be the output of the demo application?

```java
// File: MyClass.java
public class MyClass {
    protected void finalize() {
        System.out.println("The object is being garbage collected.");
    }

    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj = null;
        System.gc();
    }
}
```

a) The code will not compile because of the call to `System.gc()`.

b) The code will compile, but the `finalize()` method will not be invoked.
c) The code will compile, and "The object is being garbage collected." will be printed.
d) The code will compile, but the output will be "null."

Answer: c) The code will compile, and "The object is being garbage collected." will be printed.

3. MCQ: What is the purpose of overriding the `finalize()` method in Java?
a) To manually trigger garbage collection for a specific object.
b) To free up resources and perform cleanup before an object is garbage collected.
c) To explicitly deallocate memory occupied by an object.
d) To prevent garbage collection for a specific object.

Answer: b) To free up resources and perform cleanup before an object is garbage collected.

4. MCQ: When does the JVM call the `finalize()` method during garbage collection?
a) Immediately after an object is created.
b) Before the `System.gc()` method is called.
c) Before the JVM exits.
d) Before deallocating memory for an object.

Answer: d) Before deallocating memory for an object.

5. MCQ (Coding): What is the output of the demo application if `MyClass` is modified as follows?

```java
// File: MyClass.java
public class MyClass {
    protected void finalize() {
        System.out.println("The object is being garbage collected.");
    }

    public static void main(String[] args) {
        MyClass obj1 = new MyClass();
        MyClass obj2 = new MyClass();
        obj1 = null;
        System.gc();
    }
}
```

a) The code will not compile because of multiple instances of `MyClass`.
b) The code will compile, but the `finalize()` method will not be invoked.
c) The code will compile, and "The object is being garbage collected." will be printed twice.
d) The code will compile, and "The object is being garbage collected." will be printed once.

Answer: d

) The code will compile, and "The object is being garbage collected." will be printed once.

6. MCQ: Is it recommended to rely on the `finalize()` method for resource cleanup in Java?
a) Yes, it is the most efficient way to clean up resources.
b) Yes, it guarantees that resources are properly cleaned up before an object is garbage collected.
c) No, it is not guaranteed that the `finalize()` method will be called promptly by the JVM.
d) No, the `finalize()` method is deprecated and should not be used for resource cleanup.

Answer: c) No, it is not guaranteed that the `finalize()` method will be called promptly by the JVM.

7. MCQ: What happens if an uncaught exception is thrown inside the `finalize()` method?
a) The JVM will handle the exception, and the garbage collection process will continue.
b) The exception will be propagated to the calling code that triggered garbage collection.

c) The JVM will immediately terminate the application.
d) The JVM will skip the `finalize()` method and proceed with garbage collection.

Answer: d) The JVM will skip the `finalize()` method and proceed with garbage collection.

11. Wrapper Classes and String Class

11.1 Wrapper Classes and String Class

1. MCQ: Which of the following is NOT a wrapper class in Java?
a) Integer
b) Boolean
c) String
d) Character

Answer: c) String

2. MCQ: What is the purpose of wrapper classes in Java?
a) To convert primitive data types into objects.
b) To create instances of abstract classes.
c) To provide utility methods for strings.
d) To define custom data types.

Answer: a) To convert primitive data types into objects.

3. MCQ (Coding): Which of the following statements correctly demonstrates boxing in Java?

```java
// Option 1
int num = 42;
Integer boxedNum = new Integer(num);

// Option 2
int num = 42;
Integer boxedNum = Integer.valueOf(num);

// Option 3
Integer boxedNum = 42;
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: d) Both Option 1 and Option 2

4. MCQ: What is the difference between autoboxing and manual boxing in Java?
a) There is no difference; both terms refer to the same process.
b) Autoboxing is done automatically by the JVM, while manual boxing is done explicitly by the programmer.
c) Autoboxing is used for primitive data types, while manual boxing is used for reference data types.
d) Manual boxing is done automatically by the JVM, while autoboxing is done explicitly by the programmer.

Answer: b) Autoboxing is done automatically by the JVM, while manual boxing is done explicitly by the programmer.

5. MCQ: Can a wrapper class be null in Java?
   a) Yes, all wrapper classes can be null.
   b) No, wrapper classes cannot have a null value.
   c) Yes, except for Boolean wrapper class, all others can be null.
   d) Yes, except for Character wrapper class, all others can be null.

   Answer: c) Yes, except for Boolean wrapper class, all others can be null.

6. MCQ: What is the purpose of the `toString()` method in wrapper classes?
   a) It converts a wrapper object to its corresponding primitive data type.
   b) It converts a wrapper object to a string representation.
   c) It converts a string to a wrapper object.
   d) It converts a primitive data type to its corresponding wrapper object.

   Answer: b) It converts a wrapper object to a string representation.

7. MCQ (Coding): What is the output of the following Java code snippet?

```java
Integer num1 = 10;
Integer num2 = 10;
System.out.println(num1 == num2);
```

   a) true
   b) false
   c) Compilation error
   d) Runtime exception

   Answer: a) true

11.2 String Class, StringBuffer & StringBuilder Class

1. MCQ: Which class among `String`, `StringBuffer`, and `StringBuilder` is immutable in Java?
   a) String
   b) StringBuffer
   c) StringBuilder
   d) All of them are immutable.

   Answer: a) String

2. MCQ: What is the main difference between `StringBuffer` and `StringBuilder` in Java?
   a) `StringBuffer` is mutable, while `StringBuilder` is immutable.
   b) `StringBuffer` is faster than `StringBuilder`.
   c) `StringBuffer` is synchronized, while `StringBuilder` is not.
   d) `StringBuffer` is used for single-threaded applications, while `StringBuilder` is used for multi-threaded applications.

   Answer: c) `StringBuffer` is synchronized, while `StringBuilder` is not.

3. MCQ (Coding): Which of the following statements correctly demonstrates appending a string in `StringBuffer` and `StringBuilder`?

```java
// Option 1
StringBuffer sb = new StringBuffer();
sb.append("Hello");

// Option 2
```

```
StringBuilder sb = new StringBuilder();
sb.append("Hello");

// Option 3
StringBuffer sb = new StringBuffer("Hello");
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: d) Both Option 1 and Option 2

4. MCQ: Which of the following classes is recommended to use when multiple threads are involved?
a) String
b) StringBuilder
c) StringBuffer
d) Both StringBuilder and StringBuffer can be used interchangeably.

Answer: c) StringBuffer

5. MCQ: What happens when you concatenate strings using the `+` operator in Java?
a) The original strings are modified in-place.
b) A new string object is created in memory.
c) The `String` class uses `StringBuilder` internally for concatenation.
d) The `String` class uses `StringBuffer` internally for concatenation.

Answer: b) A new string object is created in memory.

6. MCQ: What is the advantage of using `StringBuilder` over `StringBuffer` when working with single-threaded applications?
a) `StringBuilder` is faster than `StringBuffer`.
b) `StringBuilder` is immutable, so it guarantees thread-safety.
c) `StringBuilder` is more memory-efficient.
d) `StringBuilder` provides more methods for string manipulation.

Answer: a) `StringBuilder` is faster than `StringBuffer`.

7. MCQ (Coding): What is the output of the following Java code snippet?

```java
StringBuffer sb = new StringBuffer("Hello");
sb.append(" World");
System.out.println(sb);
```

a) "Hello World"
b) "World Hello"
c) "Hello"
d) Compilation error

Answer: a) "Hello World"

11.3 String Pool

1. MCQ: What is the purpose of the string pool in Java?
a) To store string objects that are created using the `new` keyword.
b) To store frequently used strings to conserve memory and improve performance.
c) To store strings with a specific length.

d) To store strings used in multi-threaded applications.

Answer: b) To store frequently used strings to conserve memory and improve performance.

2. MCQ: Which of the following statements about the string pool is true?
a) Strings created using the `new` keyword are stored in the string pool.
b) Strings created using the `new` keyword are not stored in the string pool.
c) The string pool is a separate memory area reserved for long strings.
d) The string pool is a part of the heap memory.

Answer: b) Strings created using the `new` keyword are not stored in the string pool.

3. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str1 = "Java";
String str2 = "Java";
String str3 = new String("Java");
System.out

.println(str1 == str2);
System.out.println(str1 == str3);
```

a) true, true
b) true, false
c) false, true
d) false, false

Answer: a) true, true

4. MCQ: When does a string object get added to the string pool in Java?
a) When the string is created using the `new` keyword.
b) When the string is created using the `new` keyword and explicitly added to the pool using the `intern()` method.
c) When the string is created using string literals.
d) When the string is created using the `new` keyword and immediately assigned to a reference variable.

Answer: c) When the string is created using string literals.

5. MCQ: How can you explicitly add a string to the string pool in Java?
a) By calling the `addToPool()` method provided by the `String` class.
b) By calling the `intern()` method on the string object.
c) By converting the string to uppercase and then assigning it to a reference variable.
d) By appending the string with another string and then assigning it to a reference variable.

Answer: b) By calling the `intern()` method on the string object.

6. MCQ: What is the benefit of using the string pool in Java?
a) It allows string objects to be garbage collected more efficiently.
b) It prevents the creation of duplicate string objects with the same content.
c) It allows strings to be compared using the `==` operator instead of the `equals()` method.
d) It prevents string objects from being modified.

Answer: b) It prevents the creation of duplicate string objects with the same content.

7. MCQ (Coding): What is the output of the following Java code snippet?

```java
```

```
String str1 = new String("Hello").intern();
String str2 = new String("Hello").intern();
System.out.println(str1 == str2);
```

a) true
b) false
c) Compilation error
d) Runtime exception

Answer: a) true

11.4 Create Sample Classes to Understand Boxing & Unboxing.

1. MCQ: What is boxing in Java?
a) Converting a primitive data type into an object of the corresponding wrapper class.
b) Converting an object of the corresponding wrapper class into a primitive data type.
c) Converting a string into a wrapper object.
d) Converting a wrapper object into a string.

Answer: a) Converting a primitive data type into an object of the corresponding wrapper class.

2. MCQ (Coding): What is the output of the following Java code snippet?

```java
int num = 42;
Integer boxedNum = num;
System.out.println(boxedNum);
```

a) 42
b) "42"
c) Compilation error
d) Runtime exception

Answer: a) 42

3. MCQ: What is unboxing in Java?
a) Converting a primitive data type into an object of the corresponding wrapper class.
b) Converting an object of the corresponding wrapper class into a primitive data type.
c) Converting a string into a wrapper object.
d) Converting a wrapper object into a string.

Answer: b) Converting an object of the corresponding wrapper class into a primitive data type.

4. MCQ (Coding): What is the output of the following Java code snippet?

```java
Integer boxedNum = 25;
int num = boxedNum;
System.out.println(num);
```

a) 25
b) "25"
c) Compilation error
d) Runtime exception

Answer: a) 25

5. MCQ: Which of the following statements is true about boxing and unboxing in Java?
   a) Boxing and unboxing are both performed automatically by the JVM.
   b) Boxing is performed automatically by the JVM, but unboxing must be done explicitly by the programmer.
   c) Unboxing is performed automatically by the JVM, but boxing must be done explicitly by the programmer.
   d) Both boxing and unboxing must be done explicitly by the programmer.

   Answer: a) Boxing and unboxing are both performed automatically by the JVM.

6. MCQ (Coding): What is the output of the following Java code snippet?

```java
Integer num1 = 100;
Integer num2 = 100;
System.out.println(num1 == num2);

Integer num3 = 200;
Integer num4 = 200;
System.out.println(num3 == num4);
```

   a) true, true
   b) true, false
   c) false, true
   d) false, false

   Answer: b) true, false

7. MCQ: What happens if you try to unbox a `null` wrapper object in Java?
   a) It will result in a runtime exception.
   b) It will automatically convert to the default value of the corresponding primitive data type.
   c) It will result in a compilation error.
   d) It will return `null`.

   Answer: a) It will result in a runtime exception.

11.5 Use Different Methods of Java Defined Wrapper Classes

1. MCQ: Which wrapper class provides the `parseXXX()` method to convert a string to a primitive data type?
   a) `Integer`
   b) `Double`
   c) `Character`
   d) `Boolean`

   Answer: a) `Integer`

2. MCQ (Coding): What is the output of the following Java code snippet?

```java
int num = Integer.parseInt("42");
System.out.println(num);
```

   a) 42
   b) "42"
   c) Compilation error
   d) Runtime exception

Answer: a) 42

3. MCQ: Which wrapper class provides the `valueOf()` method to convert a string to an object of the corresponding wrapper class?
a) `Integer`
b) `Double`
c) `Character`
d) `Boolean`

Answer: b) `Double`

4. MCQ (Coding): What is the output of the following Java code snippet?

```java
Double value = Double.valueOf("3.14");
System.out.println(value);
```

a) 3.14
b) "3.14"
c) Compilation error
d) Runtime exception

Answer: a) 3.14

5. MCQ: Which wrapper class provides the `toString()` method to convert a primitive data type to its corresponding string representation?
a) `Integer`
b) `Double`
c) `Character`
d) `Boolean`

Answer: d) `Boolean`

6. MCQ (Coding): What is the output of the following Java code snippet?

```java
boolean flag = true;
String str = Boolean.toString(flag);
System.out.println(str);
```

a) true
b) "true"
c) Compilation error
d) Runtime exception

Answer: b) "true"

7. MCQ: Which wrapper class provides the `compare()` method to compare two primitive values or wrapper objects?
a) `Integer`
b) `Double`
c) `Character`
d) `Boolean`

Answer: a) `Integer`

11.6 Create StringDemo Class and Perform Different String Manipulation Methods.

8. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "Java Programming";
String upperCaseStr = str.toUpperCase();
System.out.println(upperCaseStr);
```

a) "java programming"
b) "JAVA PROGRAMMING"
c) "Java Programming"
d) "Java programming"

Answer: b) "JAVA PROGRAMMING"

9. MCQ: What is the purpose of the `toLowerCase()` method in the `String` class?
a) To convert the entire string to lowercase.
b) To convert the entire string to uppercase.
c) To convert the first character of the string to lowercase.
d) To convert the first character of each word in the string to lowercase.

Answer: a) To convert the entire string to lowercase.

10. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "Hello, World!";
String lowerCaseStr = str.toLowerCase();
System.out.println(lowerCaseStr);
```

a) "HELLO, WORLD!"
b) "hello, world!"
c) "Hello, World!"
d) "hello, world!"

Answer: b) "hello, world!"

11. MCQ: What is the purpose of the `trim()` method in the `String` class?
a) To remove all leading and trailing whitespaces from the string.
b) To remove all leading whitespaces from the string.
c) To remove all trailing whitespaces from the string.
d) To remove all spaces from the string.

Answer: a) To remove all leading and trailing whitespaces from the string.

12. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "  Hello, World!   ";
String trimmedStr = str.trim();
System.out.println(trimmedStr);
```

a) "Hello, World!"
b) "Hello, World!   "
c) " Hello, World!"
d) " Hello, World!   "

Answer: a) "Hello, World!"

13. MCQ: What is the purpose of the `replace()` method in the `String` class?
   a) To replace all occurrences of a specific character with another character.
   b) To replace the first occurrence of a specific character with another character.
   c) To replace all occurrences of a specific substring with another substring.
   d) To replace the first occurrence of a specific substring with another substring.

   Answer: c) To replace all occurrences of a specific substring with another substring.

14. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "Hello, World!";
String replacedStr = str.replace("l", "X");
System.out.println(replacedStr);
```

   a) "HexXo, WorXd!"
   b) "Hello, World!"
   c) "HeXXo, WorXd!"
   d) "HeXXo, WorXd!X"

   Answer: a) "HexXo, WorXd!"

15. MCQ: What is the purpose of the `startsWith()` method in the `String` class?
   a) To check if the string starts with a specific character.
   b) To check if the string starts with a specific substring.
   c) To check if the string ends with a specific character.
   d) To check if the string ends with a specific substring.

   Answer: b) To check if the string starts with a specific substring.

16. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "Hello, World!";
boolean startsWithHello = str.startsWith("Hello");
boolean startsWithHi = str.startsWith("Hi");
System.out.println(startsWithHello);
System.out.println(startsWithHi);
```

   a) true, false
   b) false, true
   c) true, true
   d) false, false

   Answer: a) true, false

17. MCQ: What is the purpose of the `endsWith()` method in the `String` class?
   a) To check if the string starts with a specific character.
   b) To check if the string starts with a specific substring.
   c) To check if the string ends with a specific character.
   d) To check if the string ends with a specific substring.

   Answer: c) To check if the string ends with a specific character.

18. MCQ (Coding): What is the output of the following Java code snippet?

```java
```

```
String str = "Hello, World!";
boolean endsWithWorld = str.endsWith("World!");
boolean endsWithJava = str.endsWith("Java");
System.out.println(endsWithWorld);
System.out.println(endsWithJava);
```

a) true, false
b) false, true
c) true, true
d) false, false

Answer: a) true, false

19. MCQ: What is the purpose of the `contains()` method in the `String` class?
a) To check if the string contains a specific character.
b) To check if the string contains a specific substring.
c) To check if the string contains a specific word.
d) To check if the string contains any numeric digits.

Answer: b) To check if the string contains a specific substring.

20. MCQ (Coding): What is the output of the following Java code snippet?

```java
String str = "Hello, World!";
boolean containsHello = str.contains("Hello");
boolean containsHi = str.contains("Hi");
System.out.println(containsHello);
System.out.println(containsHi);
```

a) true, false
b) false, true
c) true, true
d) false, false

Answer: a) true, false

12. Lecture: Exception Handling

12.1 Exception Hierarchy, Errors, Checked and Unchecked Exceptions

1. MCQ: Which class is at the top of the Java Exception hierarchy?
a) `RuntimeException`
b) `Exception`
c) `Error`
d) `Throwable`

Answer: d) `Throwable`

2. MCQ: What is the key difference between checked and unchecked exceptions in Java?
a) Checked exceptions are explicitly handled by the programmer, while unchecked exceptions are handled by the JVM.
b) Checked exceptions are always related to syntax or compilation errors, while unchecked exceptions are runtime errors.

c) Checked exceptions are subclasses of `Exception`, while unchecked exceptions are subclasses of `RuntimeException`.
d) Checked exceptions require explicit handling using try-catch or throws, while unchecked exceptions do not.

Answer: d) Checked exceptions require explicit handling using try-catch or throws, while unchecked exceptions do not.

3. MCQ (Coding): Which of the following statements correctly demonstrates an unchecked exception in Java?

```java
// Option 1
int result = 10 / 0;

// Option 2
String str = null;
int length = str.length();

// Option 3
int[] arr = { 1, 2, 3 };
int value = arr[5];
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 3

Answer: d) Both Option 1 and Option 3

4. MCQ: What are Errors in Java?
a) Errors are exceptions that occur due to programming errors or invalid input.
b) Errors are runtime issues that are caught and handled by the programmer.
c) Errors are critical problems that cannot be recovered, and usually, the application should terminate.
d) Errors are another name for checked exceptions.

Answer: c) Errors are critical problems that cannot be recovered, and usually, the application should terminate.

5. MCQ: Which of the following statements about the `RuntimeException` class is true?
a) `RuntimeException` is a checked exception.
b) `RuntimeException` is an unchecked exception.
c) `RuntimeException` is an Error.
d) `RuntimeException` is a subclass of `Exception`.

Answer: b) `RuntimeException` is an unchecked exception.

6. MCQ: When should you use checked exceptions in your code?
a) Checked exceptions should be used for all possible errors, even minor ones.
b) Checked exceptions should be used for critical issues that might lead to the termination of the application.
c) Checked exceptions should be used only for issues that can be reasonably handled by the calling code.
d) Checked exceptions should not be used in Java.

Answer: c) Checked exceptions should be used only for issues that can be reasonably handled by the calling code.

7. MCQ (Coding): What is the output of the following Java code snippet?

```java
public void performOperation() throws IOException {
    // Some code that may throw an IOException
}

public static void main(String[] args) {
    try {
        performOperation();
        System.out.println("Operation successful.");
    } catch (IOException e) {
        System.out.println("An error occurred: " + e.getMessage());
    }
}
```

a) Operation successful.
b) An error occurred: (the error message)
c) Compilation error
d) Runtime exception

Answer: b) An error occurred: (the error message)

12.2 Exception Propagation

1. MCQ: What is exception propagation in Java?
a) Exception propagation is the process of throwing an exception from a method to its calling method.
b) Exception propagation is the process of catching an exception in a try-catch block.
c) Exception propagation is the process of handling checked exceptions.
d) Exception propagation is the process of converting an unchecked exception to a checked exception.

Answer: a) Exception propagation is the process of throwing an exception from a method to its calling method.

2. MCQ (Coding): What will happen if an exception is thrown from a method, and the method does not have a try-catch block to handle it?

```java
public void methodA() {
    throw new RuntimeException("Exception in methodA");
}

public void methodB() {
    methodA();
}
```

a) The program will terminate abruptly with an error message.
b) The exception will be caught and handled by the JVM.
c) The exception will be caught and handled by methodB.
d) The exception will propagate up the call stack until it is caught or the program terminates.

Answer: d) The exception will propagate up the call stack until it is caught or the program terminates.

3. MCQ: What happens if an exception is not caught by any method in the call stack?
a) The program will continue to execute normally without any impact.
b) The program will terminate abruptly with an error message.
c) The JVM will handle the exception automatically.
d) The method where the exception occurred will retry the operation.

Answer: b) The program will terminate abruptly with an error message.

4. MCQ (Coding): What will be the output of the following Java code snippet?

```java
public void methodA() {
    throw new ArithmeticException("Exception in methodA");
}

public void methodB() {
    methodA();
}

public void methodC() {
    try {
        methodB();
    } catch (RuntimeException e) {
        System.out.println("Caught: " + e.getMessage());
    }
}

public static void main(String[] args) {
    new ExceptionPropagationDemo().methodC();
}
```

a) The program will terminate abruptly with an error message.
b) The output will be "Caught: Exception in methodA".
c) The output will be "Caught: Exception in methodB".
d) The output will be "Caught: Exception in methodC".

Answer: b) The output will be "Caught: Exception in methodA".

5. MCQ: Which of the following keywords can be used to propagate an exception explicitly?
a) `catch`
b) `throws`
c) `try`
d) `finally`

Answer: b) `throws`

6. MCQ (Coding): Which of the following method declarations correctly indicates that the method may throw multiple exceptions?

```java
// Option 1
public void myMethod() throws IOException, NullPointerException {
    // Method implementation
}

// Option 2
public void myMethod() throws Exception {
    // Method implementation
}

// Option 3
public void myMethod() {
    try {
        // Method implementation
    } catch (IOException | NullPointerException e) {
```

```
        // Exception handling
    }
}
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: a) Option 1

7. MCQ: What is the advantage of using exception propagation in Java?
a) It allows the program to recover from unexpected errors and continue execution.
b) It simplifies the error-handling process by automatically catching all exceptions.
c) It helps in minimizing code duplication by reusing existing exception-handling logic.
d) It

provides a way to pass information about the error from the method where it occurred to the caller.

Answer: d) It provides a way to pass information about the error from the method where it occurred to the caller.

12.3 try-catch-finally Block, throws Clause, and throw Keyword

1. MCQ: Which keyword is used to specify a block of code where an exception may occur in Java?
a) `try`
b) `catch`
c) `finally`
d) `throws`

Answer: a) `try`

2. MCQ (Coding): What is the purpose of the `catch` block in a try-catch-finally statement?

```java
try {
    // Code that may throw an exception
} catch (ExceptionType e) {
    // Code to handle the exception
}
```

a) To define the code that will be executed if an exception of type `ExceptionType` occurs.
b) To define the code that will be executed whether an exception occurs or not.
c) To define the code that will be executed if no exception occurs.
d) To define the code that will be executed if an exception occurs, regardless of its type.

Answer: d) To define the code that will be executed if an exception occurs, regardless of its type.

3. MCQ: What is the purpose of the `finally` block in a try-catch-finally statement?
a) To define the code that will be executed if an exception occurs.
b) To define the code that will be executed whether an exception occurs or not.
c) To define the code that will be executed if no exception occurs.
d) To define the code that will be executed after the `catch` block is executed.

Answer: b) To define the code that will be executed whether an exception occurs or not.

4. MCQ (Coding): What is the output of the following Java code snippet?

```java
public void divide(int num1, int num2) {
    try {
        int result = num1 / num2;
        System.out.println("Result: " + result);
    } catch (ArithmeticException e) {
        System.out.println("Error: " + e.getMessage());
    } finally {
        System.out.println("Finally block executed.");
    }
}

public static void main(String[] args) {
    new ExceptionHandlingDemo().divide(10, 0);
}
```

a) Result: Infinity
   Finally block executed.
b) Error: / by zero
   Finally block executed.
c) Compilation error
d) Runtime exception

Answer: b) Error: / by zero
        Finally block executed.

5. MCQ: What is the purpose of the `throws` clause in a method declaration?
a) To specify the type of exceptions that the method may throw.
b) To specify the type of exceptions that the method can catch and handle.
c) To specify the type of exceptions that the method will catch and rethrow.
d) To specify the type of exceptions that the method will catch and suppress.

Answer: a) To specify the type of exceptions that the method may throw.

6. MCQ (Coding): Which of the following method declarations is correctly using the `throws` clause?

```java
// Option 1
public void myMethod() throws IOException {
    // Method implementation
}

// Option 2
public void myMethod() throws Exception {
    // Method implementation
}

// Option 3
public void myMethod() {
    try {
        // Method implementation
    } catch (IOException e) {
        // Exception handling
    }
}
```

a) Option 1
b) Option 2

c) Option 3
d) Both Option 1 and Option 2

Answer: a) Option 1

7. MCQ: What is the purpose of the `throw` keyword in Java?
a) To throw a custom exception created by the programmer.
b) To catch and handle exceptions.
c) To specify the type of exceptions that a method may throw.
d) To execute a block of code whether an exception occurs or not.

Answer: a) To throw a custom exception created by the programmer.

12.4 Multi-Catch Block

1. MCQ: What is the purpose of the multi-catch block introduced in Java 7?
a) To handle multiple exceptions of different types in a single catch block.
b) To handle multiple exceptions of the same type in a single catch block.
c) To handle exceptions thrown by multiple threads.
d) To handle exceptions thrown by multiple methods.

Answer: a) To handle multiple exceptions of different types in a single catch block.

2. MCQ (Coding): Which of the following code snippets correctly uses a multi-catch block?

```java
// Option 1
try {
    // Some code that may throw exceptions
} catch (IOException | NullPointerException e) {
    // Exception handling
}

// Option 2
try {
    // Some code that may throw exceptions
} catch (IOException e1 | NullPointerException e2) {
    // Exception handling
}

// Option 3
try {
    // Some code that may throw exceptions
} catch (Exception e) {
    // Exception handling
}
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: a) Option 1

3. MCQ: What happens if multiple exception types in a multi-catch block have an inheritance relationship?
a) The code will not compile due to ambiguity.
b) The catch block will only catch exceptions of the superclass type, not the subclasses.
c) The catch block will only catch exceptions of the subclass type, not the superclass.

d) The code will compile, and the catch block will catch exceptions of both the superclass and subclass types.

Answer: d) The code will compile, and the catch block will catch exceptions of both the superclass and subclass types.

4. MCQ (Coding): What is the output of the following Java code snippet?

```java
try {
    int[] arr = { 1, 2, 3 };
    int value = arr[5];
} catch (ArrayIndexOutOfBoundsException | ArithmeticException e) {
    System.out.println("Caught: " + e.getClass().getSimpleName());
}
```

a) Caught: ArrayIndexOutOfBoundsException
b) Caught: ArithmeticException
c) Caught: Exception
d) Caught: RuntimeException

Answer: a) Caught: ArrayIndexOutOfBoundsException

5. MCQ: In which situations is using a multi-catch block beneficial in Java?
a) When the catch blocks have identical exception handling code.
b) When the catch blocks have different exception handling code for each exception.
c) When the try block has a single statement.
d) When the try block has multiple statements.

Answer: a) When the catch blocks have identical exception handling code.

6. MCQ (Coding): Which of the following code snippets will cause a compilation error?

```java
// Option 1
try {
    // Some code that may throw exceptions
} catch (IOException | NullPointerException e) {
    // Exception handling


}

// Option 2
try {
    // Some code that may throw exceptions
} catch (Exception e1 | RuntimeException e2) {
    // Exception handling
}

// Option 3
try {
    // Some code that may throw exceptions
} catch (NullPointerException e) {
    // Exception handling
} catch (IOException e) {
    // Exception handling
}
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: b) Option 2

7. MCQ: What is the benefit of using a multi-catch block over separate catch blocks for each exception type?
a) It reduces the number of catch blocks required in the code.
b) It improves the performance of exception handling.
c) It simplifies the error-handling process and makes the code more concise.
d) It allows the program to recover from any type of exception thrown.

Answer: c) It simplifies the error-handling process and makes the code more concise.

12.5 Creating User-Defined Checked and Unchecked Exceptions

1. MCQ: In Java, what is a user-defined exception?
a) An exception that is thrown by the JVM.
b) An exception that is predefined in the Java standard library.
c) An exception that is defined by the programmer by extending the `Exception` class.
d) An exception that is automatically handled by the `catch` block.

Answer: c) An exception that is defined by the programmer by extending the `Exception` class.

2. MCQ (Coding): Which of the following code snippets correctly defines a user-defined checked exception?

```java
// Option 1
class MyCheckedException extends RuntimeException {
    // Constructor and other methods
}

// Option 2
class MyCheckedException extends Exception {
    // Constructor and other methods
}

// Option 3
class MyCheckedException extends Error {
    // Constructor and other methods
}
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: b) Option 2

3. MCQ: What is the key difference between a checked and an unchecked user-defined exception?
a) Checked exceptions must be explicitly caught or declared with `throws`, while unchecked exceptions do not require explicit handling.
b) Unchecked exceptions must be explicitly caught or declared with `throws`, while checked exceptions do not require explicit handling.

c) Checked exceptions are always related to syntax or compilation errors, while unchecked exceptions are runtime errors.
d) Checked exceptions are subclasses of `RuntimeException`, while unchecked exceptions are subclasses of `Exception`.

Answer: a) Checked exceptions must be explicitly caught or declared with `throws`, while unchecked exceptions do not require explicit handling.

4. MCQ (Coding): Which of the following code snippets correctly defines a user-defined unchecked exception?

```java
// Option 1
class MyUncheckedException extends RuntimeException {
    // Constructor and other methods
}

// Option 2
class MyUncheckedException extends Exception {
    // Constructor and other methods
}

// Option 3
class MyUncheckedException extends Error {
    // Constructor and other methods
}
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: a) Option 1

5. MCQ: When should you create a user-defined checked exception in Java?
a) When you want to handle a critical error that cannot be recovered and requires the application to terminate.
b) When you want to indicate exceptional conditions that may occur during the execution of your code.
c) When you want to handle exceptions that can be easily recovered and do not impact the normal flow of the program.
d) When you want to create an exception that does not require explicit handling.

Answer: b) When you want to indicate exceptional conditions that may occur during the execution of your code.

6. MCQ (Coding): Which of the following code snippets demonstrates the correct way to throw a user-defined exception?

```java
// Option 1
throw new MyCheckedException("This is a checked exception.");

// Option 2
throw new MyUncheckedException("This is an unchecked exception.");
```

a) Option 1
b) Option 2

c) Both Option 1 and Option 2
d) Neither Option 1 nor Option 2

Answer: c) Both Option 1 and Option 2

7. MCQ: What should you consider when designing custom exceptions in Java?
a) Custom exceptions should always extend the `RuntimeException` class.
b) Custom exceptions should always be defined as `final` to prevent subclassing.
c) Custom exceptions should have meaningful names that indicate the specific exceptional condition.
d) Custom exceptions should override the `getMessage()` method to provide more context about the exception.

Answer: c) Custom exceptions should have meaningful names that indicate the specific exceptional condition.

12.6 Create User-Defined Checked and Unchecked Exceptions.
1. MCQ: What is the purpose of creating a user-defined checked exception?
a) To indicate exceptional conditions that may occur during the execution of the code.
b) To handle a critical error that cannot be recovered and requires the application to terminate.
c) To create an exception that does not require explicit handling.
d) To demonstrate the use of inheritance in exception classes.

Answer: a) To indicate exceptional conditions that may occur during the execution of the code.

2. MCQ (Coding): Which of the following code snippets demonstrates the correct way to create a user-defined checked exception named `CustomCheckedException`?

a) Option 1

```java
class CustomCheckedException extends RuntimeException {
    public CustomCheckedException(String message) {
        super(message);
    }
}
```

b) Option 2

```java
class CustomCheckedException extends Exception {
    public CustomCheckedException(String message) {
        super(message);
    }
}
```

c) Option 3

```java
class CustomCheckedException extends Error {
    public CustomCheckedException(String message) {
        super(message);
    }
}
```

Answer: b) Option 2

3. MCQ: What is the purpose of creating a user-defined unchecked exception?

a) To indicate exceptional conditions that may occur during the execution of the code.
b) To handle a critical error that cannot be recovered and requires the application to terminate.
c) To create an exception that does not require explicit handling.
d) To demonstrate the use of inheritance in exception classes.

Answer: a) To indicate exceptional conditions that may occur during the execution of the code.

4. MCQ (Coding): Which of the following code snippets demonstrates the correct way to create a user-defined unchecked exception named `CustomUncheckedException`?

a) Option 1

```java
class CustomUncheckedException extends RuntimeException {
    public CustomUncheckedException(String message) {
        super(message);
    }
}
```

b) Option

2

```java
class CustomUncheckedException extends Exception {
    public CustomUncheckedException(String message) {
        super(message);
    }
}
```

c) Option 3

```java
class CustomUncheckedException extends Error {
    public CustomUncheckedException(String message) {
        super(message);
    }
}
```

Answer: a) Option 1

5. MCQ (Coding): Write Java code to demonstrate how to use the user-defined checked exception `CustomCheckedException` and the user-defined unchecked exception `CustomUncheckedException` in a try-catch block.

(Note: The actual coding for this MCQ will be provided here)

```java
// Option 1
class CustomCheckedException extends Exception {
    public CustomCheckedException(String message) {
        super(message);
    }
}

// Option 2
class CustomUncheckedException extends RuntimeException {
```

```
    public CustomUncheckedException(String message) {
        super(message);
    }
}

public class ExceptionDemo {
    public static void main(String[] args) {
        try {
            // Some code that may throw CustomCheckedException
            // or CustomUncheckedException
        } catch (CustomCheckedException e) {
            // Handle CustomCheckedException
        } catch (CustomUncheckedException e) {
            // Handle CustomUncheckedException
        }
    }
}
```

a) Option 1
b) Option 2
c) Both Option 1 and Option 2
d) Neither Option 1 nor Option 2

Answer: c) Both Option 1 and Option 2

6. MCQ: What is the benefit of using user-defined exceptions in Java?
a) User-defined exceptions simplify the error-handling process and make the code more concise.
b) User-defined exceptions allow the programmer to define exceptional conditions specific to the application's context.
c) User-defined exceptions automatically handle all possible errors in the code.
d) User-defined exceptions improve the performance of the application.

Answer: b) User-defined exceptions allow the programmer to define exceptional conditions specific to the application's context.

7. MCQ (Coding): Write Java code for a user-defined checked exception named `InvalidInputException` and a user-defined unchecked exception named `DataNotFoundException`.

(Note: The actual coding for this MCQ will be provided here)

```java
class InvalidInputException extends Exception {
    public InvalidInputException(String message) {
        super(message);
    }
}

class DataNotFoundException extends RuntimeException {
    public DataNotFoundException(String message) {
        super(message);
    }
}
```

13. Lecture: 14.

13.1 Brief Introduction to InputStream, OutputStream, Reader and Writer Interfaces

1. MCQ: Which Java interfaces are used for handling input operations for binary data?
   a) `InputStream` and `OutputStream`
   b) `Reader` and `Writer`
   c) `Serializable` and `Deserialize`
   d) `BufferedReader` and `BufferedWriter`

   Answer: a) `InputStream` and `OutputStream`

2. MCQ (Coding): What is the correct way to read data from a text file using `InputStream` in Java?

   a) Option 1

```java
InputStream is = new FileInputStream("data.txt");
int data;
while ((data = is.read()) != -1) {
    System.out.print((char) data);
}
is.close();
```

   b) Option 2

```java
InputStream is = new InputStreamReader(new FileInputStream("data.txt"));
int data;
while ((data = is.read()) != -1) {
    System.out.print((char) data);
}
is.close();
```

   c) Option 3

```java
InputStream is = new FileReader("data.txt");
int data;
while ((data = is.read()) != -1) {
    System.out.print((char) data);
}
is.close();
```

   Answer: a) Option 1

3. MCQ: Which Java interfaces are used for handling input operations for character data?
   a) `InputStream` and `OutputStream`
   b) `Reader` and `Writer`
   c) `Serializable` and `Deserialize`
   d) `BufferedReader` and `BufferedWriter`

   Answer: b) `Reader` and `Writer`

4. MCQ (Coding): What is the correct way to write data to a text file using `Writer` in Java?

   a) Option 1

```java
Writer writer = new FileWriter("data.txt");
```

```
String data = "Hello, World!";
writer.write(data);
writer.close();
```

b) Option 2

```java
Writer writer = new BufferedWriter(new FileWriter("data.txt"));
String data = "Hello, World!";
writer.write(data);
writer.close();
```

c) Option 3

```java
Writer writer = new PrintWriter("data.txt");
String data = "Hello, World!";
writer.write(data);
writer.close();
```

Answer: b) Option 2

13.2 NIO Package

1. MCQ: What does NIO stand for in Java?
   a) New Input/Output
   b) Non-blocking Input/Output
   c) Network Input/Output
   d) Native Input/Output

   Answer: b) Non-blocking Input/Output

2. MCQ (Coding): Which class is used to represent a buffer in Java NIO?

   a) Option 1

```java
ByteBuffer buffer = new ByteBuffer();
```

   b) Option 2

```java
Buffer buffer = Buffer.allocate(1024);
```

   c) Option 3

```java
ByteBuffer buffer = ByteBuffer.allocate(1024);
```

   Answer: c) Option 3

3. MCQ: What is the main advantage of using NIO over traditional I/O in Java?
   a) NIO provides better performance for reading and writing large amounts of data.
   b) NIO supports only blocking I/O, which simplifies the programming model.

c) NIO provides easier serialization and deserialization of objects.
d) NIO is compatible with older versions of Java.

Answer: a) NIO provides better performance for reading and writing large amounts of data.

4. MCQ (Coding): What is the correct way to read data from a file using NIO in Java?

a) Option 1

```java
FileChannel channel = new FileChannel("data.txt", "r");
ByteBuffer buffer = ByteBuffer.allocate(1024);
int bytesRead = channel.read(buffer);
```

b) Option 2

```java
FileChannel channel = new FileInputStream("data.txt").getChannel();
ByteBuffer buffer = ByteBuffer.allocate(1024);
int bytesRead = channel.read(buffer);
```

c) Option 3

```java
FileChannel channel = new RandomAccessFile("data.txt", "r").getChannel();
ByteBuffer buffer = ByteBuffer.allocate(1024);
int bytesRead = channel.read(buffer);
```

Answer: c) Option 3

13.3 Serialization and De-serialization

1. MCQ: What is the purpose of serialization in Java?
a) Serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.
b) Serialization is used to convert a byte stream into a Java object for processing.
c) Serialization is used to convert a Java object into a character stream for reading.
d) Serialization is used to convert a character stream into a Java object for writing.

Answer: a) Serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.

2. MCQ (Coding): What is the correct way to serialize an object in Java?

a) Option 1

```java
ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("data.ser"));
oos.writeObject(myObject);
oos.close();
```

b) Option 2

```java
ObjectOutputStream oos = new ObjectOutputStream(new FileWriter("data.ser"));
oos.writeObject(myObject);
```

```
oos.close();
```

c) Option 3

```java
ObjectOutputStream oos = new ObjectOutputStream(new OutputStreamWriter(new
FileOutputStream("data.ser")));
oos.writeObject(myObject);
oos.close();
```

Answer: a) Option 1

3. MCQ: What is the purpose of de-serialization in Java?
a) De-serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.
b) De-serialization is used to convert a byte stream into a Java object for processing.
c) De-serialization is used to convert a Java object into a character stream for reading.
d) De-serialization is used to convert a character stream into a Java object for writing.

Answer: b) De-serialization is used to convert a byte stream into a Java object for processing.

4. MCQ (Coding): What is the correct way to de-serialize an object in Java?

a) Option 1

```java
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("data.ser"));
MyObject myObject = (MyObject) ois.readObject();
ois.close();
```

b) Option 2

```java
ObjectInputStream ois = new ObjectInputStream(new FileReader("data.ser"));
MyObject myObject = (MyObject) ois.readObject();
ois.close();
```

c) Option 3

```java
ObjectInputStream ois = new ObjectInputStream(new InputStreamReader(new
FileInputStream("data.ser")));
MyObject myObject = (MyObject) ois.readObject();
ois.close();
```

Answer: a) Option 1

13.4 Shallow Copy and Deep Copy

1. MCQ: What is the difference between shallow copy and deep copy in Java?
a) Shallow copy duplicates the object and its references, while deep copy creates a new object with copies of all its internal objects.
b) Shallow copy creates a

new object with copies of all its internal objects, while deep copy duplicates the object and its references.
c) Shallow copy and deep copy both create new objects with copies of all their internal objects.
d) Shallow copy and deep copy both duplicate the object and its references.

Answer: a) Shallow copy duplicates the object and its references, while deep copy creates a new object with copies of all its internal objects.

2. MCQ (Coding): Which of the following code snippets demonstrates a shallow copy of an array?

```java
// Option 1
int[] sourceArray = {1, 2, 3, 4, 5};
int[] shallowCopy = sourceArray;

// Option 2
int[] sourceArray = {1, 2, 3, 4, 5};
int[] shallowCopy = new int[sourceArray.length];
System.arraycopy(sourceArray, 0, shallowCopy, 0, sourceArray.length);

// Option 3
int[] sourceArray = {1, 2, 3, 4, 5};
int[] shallowCopy = Arrays.copyOf(sourceArray, sourceArray.length);
```

a) Option 1
b) Option 2
c) Option 3
d) Both Option 1 and Option 2

Answer: a) Option 1

3. MCQ: In the context of shallow copy and deep copy, what happens to the internal objects of the original object?
a) Shallow copy creates new instances of the internal objects, while deep copy uses the same instances.
b) Shallow copy and deep copy both create new instances of the internal objects.
c) Shallow copy and deep copy both use the same instances of the internal objects.
d) Shallow copy creates references to the internal objects, while deep copy creates new instances of the internal objects.

Answer: c) Shallow copy and deep copy both use the same instances of the internal objects.

4. MCQ (Coding): Which of the following code snippets demonstrates a deep copy of an array?

```java
// Option 1
int[] sourceArray = {1, 2, 3, 4, 5};
int[] deepCopy = sourceArray.clone();

// Option 2
int[] sourceArray = {1, 2, 3, 4, 5};
int[] deepCopy = Arrays.copyOf(sourceArray, sourceArray.length);

// Option 3
int[] sourceArray = {1, 2, 3, 4, 5};
int[] deepCopy = new int[sourceArray.length];
System.arraycopy(sourceArray, 0, deepCopy, 0, sourceArray.length);
```

a) Option 1
b) Option 2
c) Option 3
d) All options create shallow copies, not deep copies.

Answer: d) All options create shallow copies, not deep copies.

13.5 Create a Demo Class to Read & Write Image/Text Files.

1. MCQ: What is the purpose of using the `FileInputStream` and `FileOutputStream` classes in Java?
a) They are used for reading and writing character data from and to files.
b) They are used for reading and writing binary data from and to files.
c) They are used for reading and writing data over the network.
d) They are used for creating new files in the file system.

Answer: b) They are used for reading and writing binary data from and to files.

2. MCQ: What is the purpose of using the `BufferedReader` and `BufferedWriter` classes in Java?
a) They are used for reading and writing character data from and to files.
b) They are used for reading and writing binary data from and to files.
c) They are used for reading and writing data over the network.
d) They are used for creating new files in the file system.

Answer: a) They are used for reading and writing character data from and to files.

3. MCQ: Which method is used to read a line of text from a file using `BufferedReader` in Java?
a) `readLine()`
b) `read()`
c) `readChar()`
d) `readText()`

Answer: a) `readLine()`

4. MCQ: Which method is used to write a line of text to a file using `BufferedWriter` in Java?
a) `writeLine()`
b) `write()`
c) `writeChar()`
d) `writeText()`

Answer: b) `write()`

13.6 Create SerializationDemo Class to Illustrate Serialization and De-serialization Process.

1. MCQ: What is the purpose of serialization in Java?
a) Serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.
b) Serialization is used to convert a byte stream into a Java object for processing.
c) Serialization is used to convert a Java object into a character stream for reading.
d) Serialization is used to convert a character stream into a Java object for writing.

Answer: a) Serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.

2. MCQ: What is the purpose of de-serialization in Java?
a) De-serialization is used to convert a Java object into a byte stream to store it in a file or send it over the network.
b) De-serialization is used to convert a byte stream into a Java object for processing.
c) De-serialization is used to convert a Java object into a character stream for reading.
d) De-serialization is used to convert a character stream into a Java object for writing.

Answer: b) De-serialization is used to convert a byte stream into a Java object for processing.

3. MCQ: Which interface must a class implement to be eligible for serialization in Java?
a) `Serializable`
b) `SerializableObject`
c) `Serialize`
d) `SerializeObject`

Answer: a) `Serializable`

4. MCQ: What is the purpose of the `transient` keyword in Java serialization?
a) The `transient` keyword indicates that a variable should not be serialized.
b) The `transient` keyword indicates that a variable should be given higher priority during serialization.
c) The `transient` keyword indicates that a variable should be serialized twice for redundancy.
d) The `transient` keyword indicates that a variable should be serialized with a delay.

Answer: a) The `transient` keyword indicates that a variable should not be serialized.

5. MCQ (Coding): How do you de-serialize an object in Java after it has been serialized and saved to a file?

a) Option 1

```java
ObjectInputStream ois = new ObjectInputStream(new FileInputStream("data.ser"));
MyObject myObject = ois.readObject();
ois.close();
```

b) Option 2

```java
ObjectInputStream ois = new ObjectInputStream(new FileReader("data.ser"));
MyObject myObject = ois.readObject();
ois.close();
```

c) Option 3

```java
ObjectInputStream ois = new ObjectInputStream(new InputStreamReader(new
FileInputStream("data.ser")));
MyObject myObject = ois.readObject();
ois.close();
```

Answer: a) Option 1

14. Lecture: Object Class & java.util Package

14.1 Date, DateTime, Calendar Class

1. MCQ: Which class is used to represent dates in Java before Java 8?
   a) `Date`
   b) `DateTime`
   c) `Calendar`
   d) `SimpleDateFormat`

   Answer: a) `Date`

2. MCQ (Coding): What is the correct way to create a `Date` object with the current date and time in Java?

   a) Option 1

```java
Date currentDate = new Date();
```

   b) Option 2

```java
Date currentDate = Date.now();
```

   c) Option 3

```java
Date currentDate = Date.today();
```

   Answer: a) Option 1

3. MCQ: Which class should be used instead of `Date` class for representing dates and times in Java 8 and later versions?
   a) `Date`
   b) `DateTime`
   c) `Calendar`
   d) `LocalDateTime`

   Answer: d) `LocalDateTime`

4. MCQ (Coding): What is the correct way to create a `LocalDateTime` object with the current date and time in Java 8?

   a) Option 1

```java
LocalDateTime currentDateTime = new LocalDateTime();
```

   b) Option 2

```java
LocalDateTime currentDateTime = LocalDateTime.now();
```

c) Option 3

```java
LocalDateTime currentDateTime = LocalDateTime.today();
```

Answer: b) Option 2

14.2 Converting Date to String and String to Date Using SimpleDateFormat Class

1. MCQ: Which class is used to format dates into strings and parse strings back into dates in Java?
a) `DateFormatter`
b) `SimpleDateFormat`
c) `DateParser`
d) `DateTimeFormatter`

Answer: b) `SimpleDateFormat`

2. MCQ (Coding): What is the correct way to format a `Date` object into a string in Java using `SimpleDateFormat`?

a) Option 1

```java
Date currentDate = new Date();
String formattedDate = currentDate.format("yyyy-MM-dd");
```

b) Option 2

```java
Date currentDate = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
String formattedDate = sdf.format(currentDate);
```

c) Option 3

```java
Date currentDate = new Date();
String formattedDate = currentDate.format(new SimpleDateFormat("yyyy-MM-dd"));
```

Answer: b) Option 2

3. MCQ: What is the purpose of the `parse()` method in the `SimpleDateFormat` class?
a) It is used to parse a `Date` object into a formatted string.
b) It is used to parse a formatted string back into a `Date` object.
c) It is used to compare two `SimpleDateFormat` objects.
d) It is used to convert a `Date` object into a timestamp.

Answer: b) It is used to parse a formatted string back into a `Date` object.

4. MCQ (Coding): What is the correct way to parse a date string into a `Date` object using `SimpleDateFormat`?

a) Option 1

```java
String dateString = "2023-07-24";
```

Date date = new Date(dateString);
```

b) Option 2

```java
String dateString = "2023-07-24";
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
Date date = sdf.parse(dateString);
```

c) Option 3

```java
String dateString = "2023-07-24";
Date date = Date.parse(dateString);
```

Answer: b) Option 2

14.3 Object Class: Overriding toString, equals & hashCode Method

1. MCQ: In Java, what is the purpose of the `toString()` method in the `Object` class?
a) It is used to convert an object into a string representation.
b) It is used to compare two objects for equality.
c) It is used to calculate a hash code for an object.
d) It is used to check if an object is an instance of a particular class.

Answer: a) It is used to convert an object into a string representation.

2. MCQ (Coding): How can you override the `toString()` method in a custom class to provide a meaningful string representation?

a) Option 1

```java
class MyClass {
    int id;
    String name;

    public String toString() {
        return "MyClass[id=" + id + ", name=" + name + "]";
    }
}
```

b) Option 2

```java
class MyClass {
    int id;
    String name;

    public String toString() {
        return id + " - " + name;
    }
}
```

c) Option 3
```

```java
class MyClass {
    int id;
    String name;

    public String toString() {
        return id + name;
    }
}
```

Answer: a) Option 1

3. MCQ: In Java, what is the purpose of the `equals()` method in the `Object` class?
   a) It is used to convert an object into a string representation.
   b) It is used to compare two objects for equality.
   c) It is used to calculate a hash code for an object.
   d) It is used to check if an object is an instance of a particular class.

Answer: b) It is used to compare two objects for equality.

4. MCQ (Coding): How can you override the `equals()` method in a custom class to provide custom equality comparison?

a) Option 1

```java
class MyClass {
    int id;
    String name;

    public boolean equals(Object obj) {
        if (obj == this) {
            return true;
        }

        if (!(obj instanceof MyClass)) {
            return false;
        }

        MyClass other = (MyClass) obj;
        return this.id == other.id && this.name.equals(other.name);
    }
}
```

b) Option 2

```java
class MyClass {
    int id;
    String name;

    public boolean equals(MyClass other) {
        return this.id == other.id && this.name.equals(other.name);
    }
}
```

c) Option 3

```java
class MyClass {
    int id;
    String name;

    public boolean equals(Object obj) {
        return true;
    }
}
```

Answer: a) Option 1

14.4 Collections

1. MCQ: In Java, which interface is the root interface of the Java Collections Framework?
   a) `Collection`
   b) `List`
   c) `Set`
   d) `Map`

   Answer: a) `Collection`

2. MCQ (Coding): What is the correct way to create a list of strings in Java using the `ArrayList` class?

   a) Option 1

```java
ArrayList<String> myList = new ArrayList<>();
```

   b) Option 2

```java
ArrayList myList = new ArrayList<String>();
```

   c) Option 3

```java
List<String> myList = new ArrayList<>();
```

   Answer: c) Option 3

3.

MCQ: Which interface in the Java Collections Framework does not allow duplicate elements?
   a) `Collection`
   b) `List`
   c) `Set`
   d) `Map`

   Answer: c) `Set`

4. MCQ (Coding): What is the correct way to create a set of integers in Java using the `HashSet` class?

   a) Option 1

```java
HashSet<Integer> mySet = new HashSet<>();
```

b) Option 2

```java
HashSet mySet = new HashSet<Integer>();
```

c) Option 3

```java
Set<Integer> mySet = new HashSet<>();
```

Answer: a) Option 1

5. MCQ: Which interface in the Java Collections Framework is used to store key-value pairs?
a) `Collection`
b) `List`
c) `Set`
d) `Map`

Answer: d) `Map`

6. MCQ (Coding): What is the correct way to create a map of strings to integers in Java using the `HashMap` class?

a) Option 1

```java
HashMap<String, Integer> myMap = new HashMap<>();
```

b) Option 2

```java
HashMap myMap = new HashMap<String, Integer>();
```

c) Option 3

```java
Map<String, Integer> myMap = new HashMap<>();
```

Answer: c) Option 3

15. Lecture: Introduction to Collections: Collection Hierarchy

15.1 List, Queue, Set, and Map Collections

1. MCQ: Which Java interface is used to represent a collection that maintains an ordered sequence of elements with duplicates allowed?
   a) `List`
   b) `Queue`
   c) `Set`
   d) `Map`

   Answer: a) `List`

2. MCQ (Coding): Which Java class is used to implement a list that stores elements in an array and allows dynamic resizing?

   a) Option 1

```java
ArrayList<T> myList = new ArrayList<>();
```

   b) Option 2

```java
LinkedList<T> myList = new LinkedList<>();
```

   c) Option 3

```java
Vector<T> myList = new Vector<>();
```

   Answer: a) Option 1

3. MCQ: Which Java interface is used to represent a collection that maintains an ordered sequence of elements with no duplicates allowed?
   a) `List`
   b) `Queue`
   c) `Set`
   d) `Map`

   Answer: c) `Set`

4. MCQ (Coding): Which Java interface is used to implement a collection that stores elements in a queue (first-in, first-out) manner?

   a) Option 1

```java
List<T> myQueue = new List<>();
```

   b) Option 2

```java
Queue<T> myQueue = new Queue<>();
```

```

c) Option 3

```java
Queue<T> myQueue = new LinkedList<>();
```

Answer: c) Option 3

5. MCQ: Which Java interface is used to represent a collection that stores key-value pairs?

a) `List`
b) `Queue`
c) `Set`
d) `Map`

Answer: d) `Map`

15.2 List Collection:

15.2.1 ArrayList, LinkedList

1. MCQ: What is the main difference between `ArrayList` and `LinkedList` in Java?
a) `ArrayList` is a resizable array-based implementation, while `LinkedList` is a doubly-linked list-based implementation.
b) `ArrayList` allows null elements, while `LinkedList` does not allow null elements.
c) `ArrayList` is a thread-safe collection, while `LinkedList` is not thread-safe.
d) `ArrayList` allows constant-time insertion and deletion, while `LinkedList` allows constant-time search.

Answer: a) `ArrayList` is a resizable array-based implementation, while `LinkedList` is a doubly-linked list-based implementation.

2. MCQ (Coding): What is the correct way to create an `ArrayList` of strings in Java?

a) Option 1

```java
ArrayList<String> myList = new ArrayList<>();
```

b) Option 2

```java
ArrayList myList = new ArrayList<String>();
```

c) Option 3

```java
List<String> myList = new ArrayList<>();
```

Answer: a) Option 1

3. MCQ: What is the main advantage of using `LinkedList` over `ArrayList` in Java?
a) `LinkedList` allows constant-time access to elements by index.
b) `LinkedList` allows constant-time insertion and deletion.
c) `LinkedList` uses less memory compared to `ArrayList`.
```

d) `LinkedList` allows duplicate elements.

Answer: b) `LinkedList` allows constant-time insertion and deletion.

4. MCQ (Coding): What is the correct way to create a `LinkedList` of integers in Java?

a) Option 1

```java
LinkedList<Integer> myList = new LinkedList<>();
```

b) Option 2

```java
LinkedList myList = new LinkedList<Integer>();
```

c) Option 3

```java
List<Integer> myList = new LinkedList<>();
```

Answer: a) Option 1

15.2.2 Vector (Insert, Delete, Search, Sort, Iterate, Replace Operations)

1. MCQ: Which class in Java provides a thread-safe implementation of a dynamic array similar to `ArrayList`?
a) `ArrayList`
b) `LinkedList`
c) `Vector`
d) `HashSet`

Answer: c) `Vector`

2. MCQ (Coding): What is the correct way to create a `Vector` of strings in Java?

a) Option 1

```java
Vector<String> myVector = new Vector<>();
```

b) Option 2

```java
Vector myVector = new Vector<String>();
```

c) Option 3

```java
List<String> myVector = new Vector<>();
```

Answer: a) Option 1

3. MCQ: Which method is used to insert an element at the end of a `Vector` in Java?

a) `insert()`
b) `add()`
c) `append()`
d) `push()`

Answer: b) `add()`

4. MCQ (Coding): Which method is used to delete an element at a specific index from a `Vector` in Java?

a) `delete()`
b) `remove()`
c) `erase()`
d) `pop()`

Answer: b) `remove()`

15.3 Collections Class

1. MCQ: Which class in Java provides utility methods to work with collections, such as sorting, searching, and synchronizing collections?
a) `CollectionUtils`
b) `Collections`
c) `CollectionUtil`
d) `CollectionHelper`

Answer: b) `Collections`

2. MCQ (Coding): Which method is used to sort a list of elements in ascending order using the `Collections` class in Java?

a) Option 1

```java
Collections.sort(list);
```

b) Option 2

```java
Collections.sortList(list);
```

c) Option 3

```java
list.sort();
```

Answer: a) Option 1

3. MCQ: Which method is used to find the maximum element in a collection using the `Collections` class in Java?
a) `Collections.max()`
b) `Collections.maximum()`
c) `Collections.findMax()`
d) `Collections.maximumElement()`

Answer: a) `Collections.max()`

15.4 Comparable and Comparator Interfaces

1. MCQ: In Java, which interface is used to define the natural ordering of objects in a class?
   a) `Comparable`
   b) `Comparator`
   c) `Sorting`
   d) `Orderable`

   Answer: a) `Comparable`

2. MCQ (Coding): How do you implement the `Comparable` interface in a custom class to define the natural ordering?

   a) Option 1

```java
class MyClass implements Comparable {
    // implementation of compareTo() method
}
```

   b) Option 2

```java
class MyClass implements Comparable<MyClass> {
    // implementation of compareTo() method
}
```

   c) Option 3

```java
class MyClass extends Comparable {
    // implementation of compareTo() method
}
```

   Answer: b) Option 2

3. MCQ: In Java, which interface is used to define custom sorting of objects based on different criteria?
   a) `Comparable`
   b) `Comparator`
   c) `Sorting`
   d) `Orderable`

   Answer: b) `Comparator`

4. MCQ (Coding): How do you implement the `Comparator` interface in a custom class to define custom sorting?

   a) Option 1

```java
class MyComparator implements Comparator {
    // implementation of compare() method
}
```

b) Option 2

```java
class MyComparator implements Comparator<MyClass> {
    // implementation of compare() method
}
```

c) Option 3

```java
class MyComparator extends Comparator {
    // implementation of compare() method
}
```

Answer: b) Option 2

15.5 Queue Collection

1. MCQ: Which interface in Java is used to represent a collection that stores elements in a queue (first-in, first-out) manner?
a) `List`
b) `Queue`
c) `Set`
d) `Map`

Answer: b) `Queue`

2. MCQ (Coding): What is the correct way to create a `Queue` of strings in Java?

a) Option 1

```java
Queue<String> myQueue = new Queue<>();
```

b) Option 2

```java
Queue myQueue = new Queue<String>();
```

c) Option 3

```java
Queue<String> myQueue = new LinkedList<>();
```

Answer: c) Option 3

15.6 Create DateManipulator Class to Convert String to Date, Date to String and to Find Out Number of Days Between Two Dates.

1. MCQ: What is the purpose of the `SimpleDateFormat` class in Java?

a) It is used to convert date objects into formatted strings.
b) It is used to parse formatted strings back into date objects.
c) It is used to perform date arithmetic and comparisons.

d) It is used to calculate the number of days between two dates.

Answer: a) It is used to convert date objects into formatted strings.

2. MCQ: Which method is used to convert a string into a `Date` object using `SimpleDateFormat` in Java?

a) `parse()`
b) `format()`
c) `convert()`
d) `toDate()`

Answer: a) `parse()`

3. MCQ: Which method is used to convert a `Date` object into a string using `SimpleDateFormat` in Java?

a) `parse()`
b) `format()`
c) `convert()`
d) `toString()`

Answer: b) `format()`

4. MCQ: Which Java class is used to represent a point in time and provides methods to manipulate dates and times?

a) `SimpleDateFormat`
b) `DateFormatter`
c) `DateTime`
d) `Calendar`

Answer: d) `Calendar`

15.7 Create a List of Java Defined Wrapper Classes and Perform Insert/Delete/Search/Iterate/Sort Operations.

1. MCQ: What are the Java defined wrapper classes for primitive data types?
a) `Integer`, `Float`, `Character`, `Boolean`
b) `Byte`, `Short`, `Integer`, `Long`
c) `Float`, `Double`, `Character`, `Boolean`
d) `Byte`, `Short`, `Character`, `Boolean`

Answer: b) `Byte`, `Short`, `Integer`, `Long`

2. MCQ: Which Java wrapper class is used to store integers as objects and provides useful methods for manipulation?

a) `Integer`
b) `Float`
c) `Character`
d) `Boolean`

Answer: a) `Integer`

3. MCQ: Which method is used to insert an element at the end of a list using the `add()` method in Java?

a) `insert()`
b) `addLast()`
c) `append()`

d) `add()`

Answer: d) `add()`

4. MCQ: Which method is used to delete an element at a specific index from a list using the `remove()` method in Java?

a) `delete()`
b) `remove()`
c) `erase()`
d) `deleteElement()`

Answer: b) `remove()`

15.8 Create a Collection of Employee Class and Sort Objects Using Comparable and Comparator Interfaces.

1. MCQ: Which interface is used to define the natural ordering of objects in a class for sorting?

a) `Comparable`
b) `Comparator`
c) `Sorting`
d) `Orderable`

Answer: a) `Comparable`

2. MCQ: Which method is used to sort a collection of objects that implement the `Comparable` interface?

a) `sort()`
b) `Collections.sort()`
c) `sorted()`
d) `Collections.sorted()`

Answer: b) `Collections.sort()`

3. MCQ: Which interface is used to define custom sorting of objects based on different criteria?

a) `Comparable`
b) `Comparator`
c) `Sorting`
d) `Orderable`

Answer: b) `

Comparator`

4. MCQ: Which method is used to sort a collection of objects using a custom `Comparator`?

a) `sort()`
b) `Collections.sort()`
c) `sorted()`
d) `Collections.sorted()`

Answer: a) `sort()`

15.9 Implement Queue Data Structure Using LinkedList and Queue Collection.

1. MCQ: What is the primary difference between the `Queue` interface and the `LinkedList` class?

a) `Queue` is an interface, and `LinkedList` is a class.
b) `Queue` is a class, and `LinkedList` is an interface.
c) Both `Queue` and `LinkedList` are classes.
d) Both `Queue` and `LinkedList` are interfaces.

Answer: a) `Queue` is an interface, and `LinkedList` is a class.

2. MCQ: Which interface in Java is used to represent a collection that stores elements in a queue (first-in, first-out) manner?

a) `List`
b) `Queue`
c) `Set`
d) `Map`

Answer: b) `Queue`

3. MCQ: Which method is used to insert an element at the end of a queue?

a) `enqueue()`
b) `add()`
c) `insert()`
d) `push()`

Answer: a) `enqueue()`

4. MCQ: Which method is used to remove the element at the front of a queue?

a) `dequeue()`
b) `remove()`
c) `pop()`
d) `delete()`

Answer: a) `dequeue()`

16. Lecture: Collection:

16.1 HashSet, LinkedHashSet & TreeSet Collection

1. MCQ: Which Java collection class guarantees no duplicate elements and does not maintain insertion order?
a) `HashSet`
b) `LinkedHashSet`
c) `TreeSet`
d) `ArrayList`

Answer: a) `HashSet`

2. MCQ (Coding): How do you create a `HashSet` and add elements to it in Java?

a) Option 1

```java
HashSet<String> set = new HashSet<>();
set.add("apple");
set.add("banana");
set.add("orange");
```

b) Option 2

```java
HashSet set = new HashSet();
set.put("apple");
set.put("banana");
set.put("orange");
```

c) Option 3

```java
HashSet<String> set = new HashSet<>();
set.append("apple");
set.append("banana");
set.append("orange");
```

Answer: a) Option 1

3. MCQ: Which Java collection class maintains insertion order, allowing elements to be retrieved in the order they were added?
a) `HashSet`
b) `LinkedHashSet`
c) `TreeSet`
d) `ArrayList`

Answer: b) `LinkedHashSet`

4. MCQ (Coding): How do you create a `LinkedHashSet` and add elements to it in Java?

a) Option 1

```java
LinkedHashSet<String> set = new LinkedHashSet<>();
set.add("red");
set.add("green");
set.add("blue");
```

b) Option 2

```java
LinkedHashSet set = new LinkedHashSet();
set.put("red");
set.put("green");
set.put("blue");
```

c) Option 3

```java
LinkedHashSet<String> set = new LinkedHashSet<>();
set.append("red");
set.append("green");
set.append("blue");
```

Answer: a) Option 1

5. MCQ: Which Java collection class maintains elements in sorted order (natural order or custom order) and does not allow duplicate elements?
   a) `HashSet`
   b) `LinkedHashSet`
   c) `TreeSet`
   d) `ArrayList`

   Answer: c) `TreeSet`

6. MCQ (Coding): How do you create a `TreeSet` and add elements to it in Java?

   a) Option 1

```java
TreeSet<String> set = new TreeSet<>();
set.add("dog");
set.add("cat");
set.add("lion");
```

   b) Option 2

```java
TreeSet set = new TreeSet();
set.put("dog");
set.put("cat");
set.put("lion");
```

   c) Option 3

```java
TreeSet<String> set = new TreeSet<>();
set.append("dog");
set.append("cat");
set.append("lion");
```

   Answer: a) Option 1

16.2 Backed Set Collections

1. MCQ: What is the concept of "Backed Collections" in Java?
   a) It refers to collections that automatically synchronize with other collections.
   b) It refers to collections that share the same data with other collections, but with different views.
   c) It refers to collections that use a backing store to persist data.
   d) It refers to collections that allow elements to be added only to the back of the collection.

   Answer: b) It refers to collections that share the same data with other collections, but with different views.

2. MCQ (Coding): Which method is used to create a backed `Set` from an existing `HashSet` in Java?

   a) Option 1

```java
Set<String> backedSet = new HashSet<>();
backedSet.add("apple");
backedSet.add("banana");
```

```
Set<String> set = Collections.backedSet(backedSet);
```

b) Option 2

```java
HashSet<String> backedSet = new HashSet<>();
backedSet.add("apple");
backedSet.add("banana");
Set<String> set = Collections.synchronizedSet(backedSet);
```

c) Option 3

```java
HashSet<String> backedSet = new HashSet<>();
backedSet.add("apple");
backedSet.add("banana");
Set<String> set = Collections.unmodifiableSet(backedSet);
```

Answer: a) Option 1

3. MCQ: Backed collections are useful when you want to:
a) Create an immutable collection.
b) Ensure thread safety for the collection.
c) Share the same data between multiple collections with different views.
d) Add and remove elements from the collection efficiently.

Answer: c) Share the same data between multiple collections with different views.

4. MCQ (Coding): Which method is used to create a synchronized (thread-safe) `Set` from an existing `HashSet` in Java?

a) Option 1

```java
Set<String> synchronizedSet = new HashSet<>();
synchronizedSet.add("apple");
synchronizedSet.add("banana");
Set<String> set = Collections.synchronizedSet(synchronizedSet);
```

b) Option 2

```java
HashSet<String> synchronizedSet = new HashSet<>();
synchronizedSet.add("apple");
synchronizedSet.add("banana");
Set<String> set = Collections.backedSet(synchronizedSet);
```

c) Option 3

```java
Set<String> synchronizedSet = new HashSet<>();
synchronizedSet.add("apple");
synchronizedSet.add("banana");
Set<String> set = Collections.unmodifiableSet(synchronizedSet);
```

Answer: a) Option 1

5. MCQ: Backed collections are efficient because:
   a) They automatically handle synchronization for multithreaded access.
   b) They do not require any additional memory to store duplicate elements.
   c) They allow constant-time insertion and deletion operations.
   d) They share the same underlying data, reducing memory overhead.

   Answer: d) They share the same underlying data, reducing memory overhead.

6. MCQ (Coding): Which method is used to create an unmodifiable `Set` from an existing `HashSet` in Java?

   a) Option 1

```java
Set<String> unmodifiableSet = new HashSet<>();
unmodifiableSet.add("apple");
unmodifiableSet.add("banana");
Set<String> set = Collections.unmodifiableSet(unmodifiableSet);
```

   b) Option 2

```java
HashSet<String> unmodifiableSet = new HashSet<>();
unmodifiableSet.add("apple");
unmodifiableSet.add("banana");
Set<String> set = Collections.synchronizedSet(unmodifiableSet);
```

   c) Option 3

```java
Set<String> unmodifiableSet = new HashSet<>();
unmodifiableSet.add("apple");
unmodifiableSet.add("banana");
Set<String> set = Collections.backedSet(unmodifiableSet);
```

   Answer: a) Option 1

Sure, let's continue with the MCQs for the remaining topics:

16.3 Map Collection:
16.3.1 HashTable, HashMap, LinkedHashMap & TreeMap Classes

1. MCQ: Which Java collection class is synchronized and does not allow null keys or values?
   a) `Hashtable`
   b) `HashMap`
   c) `LinkedHashMap`
   d) `TreeMap`

   Answer: a) `Hashtable`

2. MCQ (Coding): How do you create a `HashMap` and add key-value pairs to it in Java?

   a) Option 1

```java
HashMap<String, Integer> map = new HashMap<>();
map.put("apple", 1);
map.put("banana", 2);
map.put("orange", 3);
```

b) Option 2

```java
HashMap map = new HashMap();
map.put("apple", 1);
map.put("banana", 2);
map.put("orange", 3);
```

c) Option 3

```java
HashMap<String, Integer> map = new HashMap<>();
map.append("apple", 1);
map.append("banana", 2);
map.append("orange", 3);
```

Answer: a) Option 1

3. MCQ: Which Java collection class maintains insertion order of keys and allows null keys and values?
a) `Hashtable`
b) `HashMap`
c) `LinkedHashMap`
d) `TreeMap`

Answer: c) `LinkedHashMap`

4. MCQ (Coding): How do you create a `LinkedHashMap` and add key-value pairs to it in Java?

a) Option 1

```java
LinkedHashMap<String, Integer> map = new LinkedHashMap<>();
map.put("red", 1);
map.put("green", 2);
map.put("blue", 3);
```

b) Option 2

```java
LinkedHashMap map = new LinkedHashMap();
map.put("red", 1);
map.put("green", 2);
map.put("blue", 3);
```

c) Option 3

```java
LinkedHashMap<String, Integer> map = new LinkedHashMap<>();
map.append("red", 1);
```

```
map.append("green", 2);
map.append("blue", 3);
```

Answer: a) Option 1

5. MCQ: Which Java collection class sorts keys in natural order or based on a custom comparator?
   a) `Hashtable`
   b) `HashMap`
   c) `LinkedHashMap`
   d) `TreeMap`

   Answer: d) `TreeMap`

6. MCQ (Coding): How do you create a `TreeMap` and add key-value pairs to it in Java?

   a) Option 1

```java
TreeMap<String, Integer> map = new TreeMap<>();
map.put("dog", 1);
map.put("cat", 2);
map.put("lion", 3);
```

   b) Option 2

```java
TreeMap map = new TreeMap();
map.put("dog", 1);
map.put("cat", 2);
map.put("lion", 3);
```

   c) Option 3

```java
TreeMap<String, Integer> map = new TreeMap<>();
map.append("dog", 1);
map.append("cat", 2);
map.append("lion", 3);
```

   Answer: a) Option 1

16.3.2 Backed Map Collections

1. MCQ: What is the purpose of "backed map collections" in Java?
   a) They automatically synchronize map collections for multithreaded access.
   b) They provide a way to map keys to multiple values.
   c) They allow sharing the same data between multiple maps with different views.
   d) They enforce that keys and values must be of the same data type.

   Answer: c) They allow sharing the same data between multiple maps with different views.

2. MCQ (Coding): Which method is used to create a backed `Map` from an existing `HashMap` in Java?

   a) Option 1

```java
```

```
Map<String, Integer> backedMap = new HashMap<>();
backedMap.put("apple", 1);
backedMap.put("banana", 2);
Map<String, Integer> map = Collections.backedMap(backedMap);
```

   b) Option 2

```java
HashMap<String, Integer> backedMap = new HashMap<>();
backedMap.put("apple", 1);
backedMap.put("banana", 2);
Map<String, Integer> map = Collections.synchronizedMap(backedMap);
```

   c) Option 3

```java
HashMap<String, Integer> backedMap = new HashMap<>();
backedMap.put("apple", 1);
backedMap.put("banana", 2);
Map<String, Integer> map = Collections.unmodifiableMap(backedMap);
```

   Answer: a) Option 1

3. MCQ: Backed map collections are useful when you want to:
   a) Create an immutable map.
   b) Ensure thread safety for the map.
   c) Share the same data between multiple maps with different views.
   d) Ensure that keys and values are unique in the map.

   Answer: c) Share the same data between multiple maps with different views.

4. MCQ (Coding): Which method is used to create a synchronized (thread-safe) `Map` from an existing `HashMap` in Java?

   a) Option 1

```java
Map<String, Integer> synchronizedMap = new HashMap<>();
synchronizedMap.put("apple", 1);
synchronizedMap.put("banana", 2);
Map<String, Integer> map = Collections.synchronizedMap(synchronizedMap);
```

   b) Option 2

```java
HashMap<String, Integer> synchronizedMap = new HashMap<>();
synchronizedMap.put("apple", 1);
synchronizedMap.put("banana", 2);
Map<String, Integer> map = Collections.backedMap(synchronizedMap);
```

   c) Option 3

```java
Map<String, Integer> synchronizedMap = new HashMap<>();
synchronizedMap.put("apple", 1);
```

```
synchronizedMap.put("banana", 2);
Map<String, Integer> map = Collections.unmodifiableMap(synchronizedMap);
```

Answer: a) Option 1

5. MCQ: Backed map collections are efficient because:
a) They automatically handle synchronization for multithreaded access.
b) They do not require any additional memory to store key-value pairs.
c) They allow constant-time insertion and deletion operations.
d) They share the same underlying data, reducing memory overhead.

Answer: d) They share the same underlying data, reducing memory overhead.

6. MCQ (Coding): Which method is used to create an unmodifiable `Map` from an existing `HashMap` in Java?

a) Option 1

```java
Map<String, Integer> unmodifiableMap = new HashMap<>();
unmodifiableMap.put("apple", 1);
unmodifiableMap.put("banana", 2);
Map<String, Integer> map = Collections.unmodifiableMap(unmodifiableMap);
```

b) Option 2

```java
HashMap<String, Integer> unmodifiableMap = new HashMap<>();
unmodifiableMap.put("apple", 1);
unmodifiableMap

.put("banana", 2);
Map<String, Integer> map = Collections.synchronizedMap(unmodifiableMap);
```

c) Option 3

```java
Map<String, Integer> unmodifiableMap = new HashMap<>();
unmodifiableMap.put("apple", 1);
unmodifiableMap.put("banana", 2);
Map<String, Integer> map = Collections.backedMap(unmodifiableMap);
```

Answer: a) Option 1

16.4 Generics

1. MCQ: What is the purpose of generics in Java?
a) Generics allow you to create objects of generic classes that can hold any type of data.
b) Generics provide a way to perform mathematical operations on numeric data types.
c) Generics ensure that collections can only hold objects and not primitive data types.
d) Generics allow you to create classes, interfaces, and methods that can work with different data types.

Answer: d) Generics allow you to create classes, interfaces, and methods that can work with different data types.

2. MCQ (Coding): How do you declare a generic class that can work with any data type in Java?

a) Option 1

```java
class MyGenericClass<T> {
    // class implementation
}
```

b) Option 2

```java
class MyGenericClass {
    <T> // generic declaration
    // class implementation
}
```

c) Option 3

```java
class MyGenericClass {
    // class implementation
    <T> // generic declaration
}
```

Answer: a) Option 1

3. MCQ: What is the benefit of using generics in collections?
a) Generics provide a way to store primitive data types in collections.
b) Generics allow collections to work with different data types, providing type safety at compile-time.
c) Generics allow collections to hold unlimited elements without any restrictions.
d) Generics allow collections to resize dynamically based on the number of elements.

Answer: b) Generics allow collections to work with different data types, providing type safety at compile-time.

4. MCQ (Coding): How do you declare a generic method that can work with different data types in Java?

a) Option 1

```java
public void myGenericMethod<T>(T obj) {
    // method implementation
}
```

b) Option 2

```java
public <T> void myGenericMethod(T obj) {
    // method implementation
}
```

c) Option 3

```java
public void myGenericMethod(T obj) {
    // method implementation
    <T> // generic declaration
}
```

Answer: b) Option 2

5. MCQ: What is the role of the diamond operator (`<>`) in generics?

a) It is used to declare multiple generic types in a single statement.
b) It is used to instantiate a generic class without specifying the data type.
c) It is used to declare a wildcard generic type that accepts any data type.
d) It is used to indicate a class is implementing a generic interface.

Answer: b) It is used to instantiate a generic class without specifying the data type.

6. MCQ (Coding): How do you instantiate a generic class with the diamond operator in Java?

a) Option 1

```java
MyGenericClass<> myObj = new MyGenericClass<>();
```

b) Option 2

```java
MyGenericClass myObj = new MyGenericClass<>();
```

c) Option 3

```java
MyGenericClass<T> myObj = new MyGenericClass<T>();
```

Answer: b) Option 2

16.5 Concurrent Collections

1. MCQ: What is the purpose of concurrent collections in Java?
a) Concurrent collections ensure that elements are stored in sorted order.
b) Concurrent collections provide a way to perform mathematical operations on numeric data types concurrently.
c) Concurrent collections allow multiple threads to access and modify collections safely without explicit synchronization.
d) Concurrent collections allow collections to be resized dynamically based on the number of elements.

Answer: c) Concurrent collections allow multiple threads to access and modify collections safely without explicit synchronization.

2. MCQ (Coding): Which concurrent collection class in Java allows multiple threads to insert elements at the beginning and end of the collection?

a) `ConcurrentList`
b) `ConcurrentMap`
c) `ConcurrentQueue`

d) `ConcurrentDeque`

Answer: d) `ConcurrentDeque`

3. MCQ: Concurrent collections achieve thread-safety by:
a) Restricting access to the collections to a single thread at a time.
b) Using explicit synchronization on collection methods.
c) Allowing only read operations on the collections.
d) Implementing internal mechanisms to handle concurrent access and modifications.

Answer: d) Implementing internal mechanisms to handle concurrent access and modifications.

4. MCQ (Coding): Which concurrent collection class in Java allows multiple threads to access and modify elements in a thread-safe manner, similar to a hash table?

a) `

ConcurrentHashMap`
b) `ConcurrentHashSet`
c) `ConcurrentLinkedList`
d) `ConcurrentTreeMap`

Answer: a) `ConcurrentHashMap`

5. MCQ: When would you use a concurrent collection in Java?
a) When you need to perform complex mathematical operations on collection elements.
b) When you need to enforce a specific order of elements in the collection.
c) When you need to ensure thread safety for accessing and modifying collections from multiple threads.
d) When you need to store elements in a sorted order.

Answer: c) When you need to ensure thread safety for accessing and modifying collections from multiple threads.

6. MCQ (Coding): Which concurrent collection class in Java allows multiple threads to access and modify elements in a thread-safe manner, similar to a regular queue (first-in, first-out)?

a) `ConcurrentList`
b) `ConcurrentMap`
c) `ConcurrentQueue`
d) `ConcurrentDeque`

Answer: c) `ConcurrentQueue`

17. Lecture:

17.1 Multi-Threading: Thread Class and Runnable Interface

1. MCQ: Which of the following interfaces in Java is used for creating a thread?
a) `Runnable`
b) `Thread`
c) `Executor`
d) `Timer`

Answer: a) `Runnable`

2. MCQ (Coding): How do you create a thread using the `Thread` class in Java?

a) Option 1

```java
Thread myThread = new Thread();
myThread.start();
```

b) Option 2

```java
Thread myThread = new Thread(new Runnable() {
    public void run() {
        // thread logic here
    }
});
myThread.start();
```

c) Option 3

```java
Thread myThread = new Thread();
myThread.run();
```

Answer: b) Option 2

3. MCQ: Which method is used to start the execution of a thread in Java?

a) `start()`
b) `run()`
c) `execute()`
d) `begin()`

Answer: a) `start()`

4. MCQ (Coding): How do you create a thread using the `Runnable` interface in Java?

a) Option 1

```java
Runnable myRunnable = new Runnable() {
    public void run() {
        // thread logic here
    }
};
Thread myThread = new Thread(myRunnable);
myThread.start();
```

b) Option 2

```java
Runnable myRunnable = new Runnable() {
    public void run() {
        // thread logic here
    }
};
myRunnable.start();
```

c) Option 3

```java
Runnable myRunnable = new Runnable() {
    public void run() {
        // thread logic here
    }
};
myRunnable.run();
```

Answer: a) Option 1

5. MCQ: What is the purpose of the `run()` method in a thread?

a) It is called when the thread is created.
b) It is used to start the execution of a thread.
c) It contains the code that will be executed by the thread.
d) It is used to pause the execution of a thread.

Answer: c) It contains the code that will be executed by the thread.

6. MCQ (Coding): How do you define the logic for a thread using the `run()` method in Java?

a) Option 1

```java
public void run() {
    // thread logic here
}
```

b) Option 2

```java
public void start() {
    // thread logic here
}
```

c) Option 3

```java
public void execute() {
    // thread logic here
}
```

Answer: a) Option 1

7. MCQ: Which of the following statements is true about the `Runnable` interface in Java?
a) It is a class that represents a thread.
b) It is an interface that allows a class to be executed as a thread.
c) It is a method that starts a thread.
d) It is a package that contains thread-related classes.

Answer: b) It is an interface that allows a class to be executed as a thread.

8. MCQ (Coding): How do you create a thread that executes the logic defined in the `run()` method of the `MyRunnable` class?

```java
class MyRunnable implements Runnable {
    public void run() {
        // thread logic here
    }
}

public class Main {
    public static void main(String[] args) {
        MyRunnable myRunnable = new MyRunnable();
        Thread myThread = new Thread(myRunnable);
        myThread.start();
    }
}
```

Answer: Use the code provided in the question.

9. MCQ: Which method is used to check if a thread is still alive and running in Java?

a) `isRunning()`
b) `isAlive()`
c) `isExecuting()`
d) `isWorking()`

Answer: b) `isAlive()`

10. MCQ (Coding): How do you check if a thread `myThread` is still alive in Java?

a) Option 1

```java
myThread.isAlive();
```

b) Option 2

```java
Thread.isAlive(myThread);
```

c) Option 3

```java
Thread.isRunning(myThread);
```

Answer: a) Option 1

17.2 sleep, join, yield, setPriority, getPriority Methods

1. MCQ: Which method is used to pause the execution of a thread for a specific amount of time in Java?

a) `sleep()`
b) `join()`
c) `yield()`
d) `pause()`

Answer: a) `sleep()`

2. MCQ (Coding): How do you pause the execution of a thread for 1 second using the `sleep()` method in Java?

a) Option 1

```java
Thread.sleep(1000);
```

b) Option 2

```java
sleep(1000);
```

c) Option 3

```java
Thread.pause(1000);
```

Answer: a) Option 1

3. MCQ: Which method is used to wait for a thread to finish its execution in Java?

a) `sleep()`
b) `join()`
c) `yield()`
d) `wait()`

Answer: b) `join()`

4. MCQ (Coding): How do you make a thread `myThread` wait for another thread `otherThread` to finish using the `join()` method in Java?

a) Option 1

```java
myThread.wait(otherThread);
```

b) Option 2

```java
myThread.join(otherThread);
```

c) Option 3

```java
otherThread.join(myThread);
```

Answer: b) Option 2

5. MCQ: Which method is used to temporarily pause the execution of a thread to give other threads a chance to run in Java?

a) `sleep()`
b) `join()`
c) `yield()`
d) `pause()`

Answer: c) `yield()`

6. MCQ (Coding): How do you make

a thread `myThread` yield its execution to other threads using the `yield()` method in Java?

a) Option 1

```java
myThread.pause();
```

b) Option 2

```java
myThread.yield();
```

c) Option 3

```java
Thread.yield(myThread);
```

Answer: b) Option 2

7. MCQ: Which method is used to set the priority of a thread in Java?

a) `setPriority()`
b) `priority()`
c) `assignPriority()`
d) `setThreadPriority()`

Answer: a) `setPriority()`

8. MCQ (Coding): How do you set the priority of a thread `myThread` to the highest priority in Java?

a) Option 1

```java
myThread.setPriority(Thread.MIN_PRIORITY);
```

b) Option 2

```java
myThread.setPriority(Thread.MAX_PRIORITY);
```

c) Option 3

```java
myThread.setPriority(Thread.HIGH_PRIORITY);
```

Answer: b) Option 2

9. MCQ: Which method is used to get the priority of a thread in Java?

a) `getPriority()`
b) `priority()`
c) `retrievePriority()`
d) `getThreadPriority()`

Answer: a) `getPriority()`

10. MCQ (Coding): How do you get the priority of a thread `myThread` in Java?

a) Option 1

```java
myThread.getPriority();
```

b) Option 2

```java
Thread.getPriority(myThread);
```

c) Option 3

```java
myThread.priority();
```

Answer: a) Option 1

17.3 ThreadGroup Class

1. MCQ: What is the purpose of the `ThreadGroup` class in Java?
a) It is used to group threads together for easy management and control.
b) It is used to create multiple instances of a thread.
c) It is used to implement the `Runnable` interface.
d) It is used to define custom thread priority levels.

Answer: a) It is used to group threads together for easy management and control.

2. MCQ (Coding): How do you create a thread group in Java?

a) Option 1

```java
ThreadGroup group = new ThreadGroup();
```

b) Option 2

```java
ThreadGroup group = new ThreadGroup("MyGroup");
```

c) Option 3

```java
ThreadGroup group = Thread.getThreadGroup();
```

Answer: b) Option 2

3. MCQ: What is the role of the `uncaughtException()` method in the `ThreadGroup` class?

a) It is called when a thread is about to terminate due to an uncaught exception.
b) It is used to set the priority of threads in the thread group.
c) It is called when a thread is paused using the `yield()` method.
d) It is used to handle checked exceptions in a thread group.

Answer: a) It is called when a thread is about to terminate due to an uncaught exception.

4. MCQ (Coding): How do you handle uncaught exceptions in a thread group using the `uncaughtException()` method in Java?

a) Option 1

```java
public void uncaughtException(Thread t, Throwable e) {
    // exception handling code here
}
```

b) Option 2

```java
public void handleException(Thread t, Throwable e) {
    // exception handling code here
}
```

c) Option 3

```java
public void catchException(Thread t, Throwable e) {
    // exception handling code here
}
```

Answer: a) Option 1

18. Lecture

18.1 Synchronization

1. MCQ: What is the main purpose of synchronization in multi-threaded Java programs?
a) To prevent threads from running concurrently.
b) To allow threads to run concurrently without any issues.
c) To avoid thread interference and memory consistency errors.
d) To ensure threads always finish their execution in a specific order.

Answer: c) To avoid thread interference and memory consistency errors.

2. MCQ (Coding): How do you synchronize a method in Java?

a) Option 1

```java
synchronized void myMethod() {
    // method logic here
}
```

   b) Option 2

```java
void myMethod() {
    synchronized {
        // method logic here
    }
}
```

   c) Option 3

```java
synchronized(myMethod) {
    // method logic here
}
```

   Answer: a) Option 1

3. MCQ: What is the role of the `synchronized` keyword in Java?

   a) It ensures that the method can only be called by a single thread at a time.
   b) It makes the method execute faster by bypassing thread synchronization.
   c) It allows multiple threads to execute the method simultaneously.
   d) It automatically creates new threads for the synchronized method.

   Answer: a) It ensures that the method can only be called by a single thread at a time.

4. MCQ (Coding): How do you synchronize a block of code in Java?

   a) Option 1

```java
synchronized(myObject) {
    // synchronized block code here
}
```

   b) Option 2

```java
synchronized {
    // synchronized block code here
}
```

   c) Option 3

```java
myObject.synchronized {
    // synchronized block code here
}
```

Answer: a) Option 1

5. MCQ: When should you use synchronization in Java?

a) Only when working with single-threaded applications.
b) When you want to make your code run faster.
c) When multiple threads access shared resources.
d) When you want to restrict thread creation.

Answer: c) When multiple threads access shared resources.

6. MCQ (Coding): How do you synchronize a static method in Java?

a) Option 1

```java
synchronized static void myStaticMethod() {
    // method logic here
}
```

b) Option 2

```java
static synchronized void myStaticMethod() {
    // method logic here
}
```

c) Option 3

```java
synchronized void static myStaticMethod() {
    // method logic here
}
```

Answer: b) Option 2

7. MCQ: What is the purpose of using synchronization in a multi-threaded environment?

a) To improve thread performance.
b) To allow threads to execute independently.
c) To prevent race conditions and data corruption.
d) To guarantee thread execution in a particular order.

Answer: c) To prevent race conditions and data corruption.

8. MCQ (Coding): How do you synchronize access to an instance variable in Java?

a) Option 1

```java
synchronized void setVariable(int value) {
    this.variable = value;
}
```

b) Option 2

```java
void setVariable(int value) {
    synchronized(this) {
        this.variable = value;
    }
}
```

c) Option 3

```java
synchronized(this.variable) {
    this.variable = value;
}
```

Answer: b) Option 2

18.2 Deadlock

1. MCQ: What is a deadlock in Java multi-threading?

a) It is a situation where a thread is unable to proceed because it is waiting for a resource that is held by another thread, and that other thread is waiting for a resource held by the first thread.
b) It is a situation where multiple threads are executing in a non-deterministic order.
c) It is a situation where a thread is unable to proceed because it is waiting for a resource that is currently being used by another thread.
d) It is a situation where a thread is stuck in an infinite loop.

Answer: a) It is a situation where a thread is unable to proceed because it is waiting for a resource that is held by another thread, and that other thread is waiting for a resource held by the first thread.

2. MCQ (Coding): Which of the following is a common cause of deadlock in multi-threading?

a) Lack of synchronization in critical sections.
b) Excessive use of `sleep()` method.
c) Excessive use of `yield()` method.
d) Lack of thread priority settings.

Answer: a) Lack of synchronization in critical sections.

3. MCQ: How can deadlock be prevented in Java?

a) By using the `yield()` method properly.
b) By setting thread priorities appropriately.
c) By avoiding circular dependencies on resources.
d) By using the `sleep()` method instead of `wait()`.

Answer: c) By avoiding circular dependencies on resources.

4. MCQ (Coding): Which Java method is used to put a thread into a waiting state until another thread notifies it?

a) `await()`
b) `suspend()`
c) `sleep()`
d

) `wait()`

Answer: d) `wait()`

5. MCQ: What is a resource acquisition hierarchy in the context of deadlock?

a) It is a way to prioritize threads based on their resource requirements.
b) It is a strategy to break the circular wait condition among threads.
c) It is a technique to avoid priority inversion.
d) It is a way to grant exclusive access to a resource for only one thread at a time.

Answer: b) It is a strategy to break the circular wait condition among threads.

6. MCQ (Coding): Which method is used to notify a waiting thread to resume execution in Java?

a) `notify()`
b) `notifyAll()`
c) `resume()`
d) `awake()`

Answer: a) `notify()`

7. MCQ: In a deadlock situation, how can you identify which threads are involved?

a) By using the `Thread.getAllStackTraces()` method.
b) By analyzing the CPU utilization of each thread.
c) By using the `Thread.dumpStack()` method.
d) By checking the thread states using the `getState()` method.

Answer: a) By using the `Thread.getAllStackTraces()` method.

8. MCQ (Coding): How can you break a deadlock in Java?

a) By forcibly terminating one of the threads involved in the deadlock.
b) By increasing the thread priorities of the deadlock threads.
c) By using the `interrupt()` method on one of the deadlock threads.
d) By releasing the resources held by one of the deadlock threads.

Answer: d) By releasing the resources held by one of the deadlock threads.

18.3 Wait, notify, and notifyAll Methods

1. MCQ: What is the purpose of the `wait()` method in Java?

a) It is used to put a thread to sleep for a specific duration.
b) It is used to pause the execution of a thread indefinitely.
c) It is used to wait for a specific condition to be met before resuming execution.
d) It is used to terminate a thread's execution.

Answer: c) It is used to wait for a specific condition to be met before resuming execution.

2. MCQ (Coding): Which method is used to notify a waiting thread to resume execution in Java?

a) `notify()`
b) `notifyAll()`
c) `resume()`
d) `awake()`

Answer: a) `notify()`

3. MCQ: What is the purpose of the `notifyAll()` method in Java?

a) It is used to wake up all waiting threads that are waiting on the same object's monitor.
b) It is used to wake up a specific thread that is waiting on the same object's monitor.
c) It is used to put all threads to sleep for a specific duration.
d) It is used to pause the execution of all threads indefinitely.

Answer: a) It is used to wake up all waiting threads that are waiting on the same object's monitor.

4. MCQ (Coding): How do you use the `wait()` method to wait for a specific condition to be met?

a) Option 1

```java
synchronized(waitCondition) {
    while (!conditionMet) {
        waitCondition.wait();
    }
}
```

b) Option 2

```java
synchronized {
    while (!conditionMet) {
        wait();
    }
}
```

c) Option 3

```java
while (!conditionMet) {
    wait();
}
```

Answer: a) Option 1

5. MCQ: What happens when a thread calls the `wait()` method in Java?

a) The thread releases the lock on the object's monitor and waits until another thread notifies it.
b) The thread continues its execution without any delay.
c) The thread enters a waiting state for a specified duration.
d) The thread terminates its execution.

Answer: a) The thread releases the lock on the object's monitor and waits until another thread notifies it.

6. MCQ (Coding): How do you use the `notifyAll()` method to wake up all waiting threads?

a) Option 1

```java
synchronized {
    conditionMet = true;
    notifyAll();
}
```

b) Option 2

```java
synchronized(waitCondition) {
    conditionMet = true;
    waitCondition.notifyAll();
}
```

c) Option 3

```java
synchronized {
    conditionMet = true;
    wait();
}
```

Answer: b) Option 2

7. MCQ: In which class are the `wait()`, `notify()`, and `notifyAll()` methods defined in Java?

a) `Thread`
b) `Object`
c) `Runnable`
d) `Lock

`

Answer: b) `Object`

8. MCQ (Coding): When do you typically use the `wait()`, `notify()`, and `notifyAll()` methods?

a) When you want to pause a thread for a specific duration.
b) When you want to stop the execution of a thread.
c) When you want to synchronize access to shared resources.
d) When you want to wait for a specific condition to be met.

Answer: d) When you want to wait for a specific condition to be met.

18.4 Producer & Consumer Problem

1. MCQ: What is the producer-consumer problem in the context of multi-threading?

a) It is a problem where multiple threads produce the same output concurrently.
b) It is a problem where multiple threads consume the same input concurrently.
c) It is a synchronization problem where threads can interfere with each other's execution.
d) It is a deadlock situation where multiple threads are waiting for each other to release resources.

Answer: b) It is a problem where multiple threads consume the same input concurrently.

2. MCQ (Coding): How can you implement the producer-consumer problem in Java?

a) Option 1

```java
class Producer {
    synchronized void produce() {
        // producer logic here
```

```
    }
}

class Consumer {
    synchronized void consume() {
        // consumer logic here
    }
}
```

b) Option 2

```java
class Producer {
    void produce() {
        synchronized(this) {
            // producer logic here
        }
    }
}

class Consumer {
    void consume() {
        synchronized(this) {
            // consumer logic here
        }
    }
}
```

c) Option 3

```java
class Producer {
    synchronized void produce() {
        // producer logic here
    }
}

class Consumer {
    void consume() {
        synchronized(this) {
            // consumer logic here
        }
    }
}
```

Answer: a) Option 1

3. MCQ: What is the role of the `wait()` and `notify()` methods in the producer-consumer problem?

a) The `wait()` method is used by the producer to wait for the consumer to consume data, and the `notify()` method is used by the consumer to notify the producer when data is available.
b) The `wait()` method is used by the consumer to wait for the producer to produce data, and the `notify()` method is used by the producer to notify the consumer when data is available.
c) Both the `wait()` and `notify()` methods are used by the producer to wait for the consumer to consume data.
d) Both the `wait()` and `notify()` methods are used by the consumer to wait for the producer to produce data.

Answer: b) The `wait()` method is used by the consumer to wait for the producer to produce data, and the `notify()` method is used by the producer to notify the consumer when data is available.

4. MCQ (Coding): How can you implement the producer-consumer problem using the `wait()` and `notify()` methods in Java?

a) Option 1

```java
class Producer {
    synchronized void produce() {
        while (dataAvailable) {
            wait();
        }
        // produce data logic here
        notify();
    }
}

class Consumer {
    synchronized void consume() {
        while (!dataAvailable) {
            wait();
        }
        // consume data logic here
        notify();
    }
}
```

b) Option 2

```java
class Producer {
    synchronized void produce() {
        while (!dataAvailable) {
            wait();
        }
        // produce data logic here
        notifyAll();
    }
}

class Consumer {
    synchronized void consume() {
        while (dataAvailable) {
            wait();
        }
        // consume data logic here
        notifyAll();
    }
}
```

c) Option 3

```java
class Producer {
    void produce() {
```

```
        synchronized(this) {
            while (!dataAvailable) {
                this.wait();
            }
            // produce data logic here
            this.notify();
        }
    }
}

class Consumer {
    void consume() {
        synchronized(this) {
            while (dataAvailable) {
                this.wait();
            }
            // consume data logic here
            this.notify();
        }
    }
}
```

Answer: a) Option 1

5. MCQ: What is the purpose of using a shared buffer in the producer-consumer problem?

a) To store the data produced by the producer.
b) To store the data consumed by the consumer.
c) To act as a communication channel between the producer and the consumer.
d) To synchronize the execution of producer and consumer threads.

Answer: c) To act as a communication channel between the producer and the consumer.

6. MCQ (Coding): How do you ensure that the producer does not produce data when the buffer is full, and the consumer does not consume data when the buffer is empty?

a) By using a shared `boolean` variable to indicate the buffer's status.
b) By using the `wait()` and `notify()` methods to wait for the buffer to become empty or full.
c) By using the `yield()` method to pause the producer and consumer when the buffer is full or empty.
d) By using the `Thread.sleep()` method to pause the producer and consumer

when the buffer is full or empty.

Answer: b) By using the `wait()` and `notify()` methods to wait for the buffer to become empty or full.

7. MCQ: What is the main challenge in implementing the producer-consumer problem?

a) Ensuring that the producer always produces data before the consumer consumes it.
b) Ensuring that the consumer always consumes data before the producer produces more.
c) Avoiding deadlock situations between the producer and consumer threads.
d) Ensuring that the producer and consumer do not access the shared buffer at the same time.

Answer: d) Ensuring that the producer and consumer do not access the shared buffer at the same time.

8. MCQ (Coding): How do you terminate the producer-consumer system in Java after a certain condition is met?

a) By using the `Thread.stop()` method on both producer and consumer threads.

b) By setting a shared `boolean` flag and checking it periodically in both producer and consumer threads.
c) By using the `Thread.interrupt()` method on both producer and consumer threads.
d) By calling the `System.exit()` method when the desired condition is met.

Answer: b) By setting a shared `boolean` flag and checking it periodically in both producer and consumer threads.

19. Lecture:

19.1 Inner Class (Regular, Method Local, Anonymous & Static Inner Class)

1. MCQ: Which type of inner class is defined within a method of an outer class and can only be accessed within that method?

a) Regular Inner Class
b) Method Local Inner Class
c) Anonymous Inner Class
d) Static Inner Class

Answer: b) Method Local Inner Class

2. MCQ (Coding): How do you create an instance of a regular inner class in Java?

a) Option 1

```java
OuterClass.RegularInnerClass inner = new OuterClass.RegularInnerClass();
```

b) Option 2

```java
RegularInnerClass inner = new OuterClass.RegularInnerClass();
```

c) Option 3

```java
OuterClass outer = new OuterClass();
RegularInnerClass inner = outer.new RegularInnerClass();
```

Answer: c) Option 3

3. MCQ: Which type of inner class has no name and is declared and instantiated at the same time?

a) Regular Inner Class
b) Method Local Inner Class
c) Anonymous Inner Class
d) Static Inner Class

Answer: c) Anonymous Inner Class

4. MCQ (Coding): How do you create an instance of an anonymous inner class in Java?

a) Option 1

```java
OuterClass.AnonymousInnerClass inner = new OuterClass.AnonymousInnerClass() {
```

```
    // anonymous inner class implementation here
};
```

b) Option 2

```java
AnonymousInnerClass inner = new OuterClass.AnonymousInnerClass() {
    // anonymous inner class implementation here
};
```

c) Option 3

```java
OuterClass outer = new OuterClass();
AnonymousInnerClass inner = outer.new AnonymousInnerClass() {
    // anonymous inner class implementation here
};
```

Answer: b) Option 2

5. MCQ: Which type of inner class is declared as `static` and can be accessed without creating an instance of the outer class?

a) Regular Inner Class
b) Method Local Inner Class
c) Anonymous Inner Class
d) Static Inner Class

Answer: d) Static Inner Class

6. MCQ (Coding): How do you create an instance of a static inner class in Java?

a) Option 1 : OuterClass.StaticInnerClass inner = new OuterClass.StaticInnerClass();
b) Option 2 : StaticInnerClass inner = new OuterClass.StaticInnerClass();
c) Option 3 : OuterClass outer = new OuterClass()\
d) Option 4 : StaticInnerClass inner = outer.new StaticInnerClass();

Answer: a) Option 1

7. MCQ: Which type of inner class can access non-final local variables of the enclosing method?

a) Regular Inner Class
b) Method Local Inner Class
c) Anonymous Inner Class
d) Static Inner Class

Answer: c) Anonymous Inner Class

8. MCQ (Coding): What is the scope of a regular inner class in Java?

a) It can be accessed only within the method where it is declared.
b) It can be accessed within the entire outer class.
c) It can be accessed within any class in the same package as the outer class.
d) It can be accessed from any class in any package.

Answer: b) It can be accessed within the entire outer class.

9. MCQ: Which type of inner class is often used for implementing callback methods or event listeners?

a) Regular Inner Class
b) Method Local Inner Class
c) Anonymous Inner Class
d) Static Inner Class

Answer: c) Anonymous Inner Class

10. MCQ (Coding): Can a regular inner class contain static methods?

a) Yes, a regular inner class can contain static methods.
b) No, a regular inner class cannot contain static methods.
c) Only if the outer class is also declared as static.
d) Only if the outer class is an interface.

Answer: b) No, a regular inner class cannot contain static methods.

19.2 Lambda Expression

1. MCQ: What is the primary purpose of using lambda expressions in Java?

a) To create new instances of classes.
b) To implement abstract methods in interfaces.
c) To create anonymous inner classes.
d) To improve the performance of the code.

Answer: b) To implement abstract methods in interfaces.

2. MCQ (Coding): How do you write a lambda expression in Java?

a) Option 1

```java
() -> {
    // lambda expression body
};
```

b) Option 2

```java
() -> (
    // lambda expression body
)
```

c) Option 3

```java
() => {
    // lambda expression body
}
```

Answer: a) Option 1

3. MCQ: In lambda expressions, how do you represent a single parameter?

a) (param)

b) (param1, param2)
c) param
d) [param]

Answer: c) param

4. MCQ (Coding): How do you use a lambda expression to implement a functional interface with a single abstract method that takes two parameters?

a) Option 1

```java
(param1, param2) -> {
    // lambda expression body
};
```

b) Option 2

```java
(param1, param2) => {
    // lambda expression body
};
```

c) Option 3

```java
(param1, param2) -> (
    // lambda expression body
)
```

Answer: a) Option 1

5. MCQ: What is the return type of a lambda expression that represents a void method?

a) `void`
b) `null`
c) `None`
d) It is inferred by the compiler.

Answer: d) It is inferred by the compiler.

6. MCQ (Coding): How do you use a lambda expression to implement a functional interface with a single abstract method that returns a value?

a) Option 1

```java
() -> {
    // lambda expression body
    return result;
};
```

b) Option 2

```java
() => {
```

```
    // lambda expression body
    return result;
};
```

   c) Option 3

```java
() -> (
    // lambda expression body
    return result;
)
```

   Answer: a) Option 1

7. MCQ: What is the purpose of using method references in Java?

   a) To call private methods from outside the class.
   b) To call static methods from non-static contexts.
   c) To simplify lambda expressions when they call existing methods.
   d) To create references to lambda expressions.

   Answer: c) To simplify lambda expressions when they call existing methods.

8. MCQ (Coding): Which of the following is a valid method reference in Java?

   a) `Math::random`
   b) `System.out::println`
   c) `String::length`
   d) All of the above

   Answer: d) All of the above

19.3 Reflection

1. MCQ: What is Reflection in Java?

   a) The process of converting an object to a byte stream.
   b) The process of analyzing and modifying the runtime behavior of classes and objects.
   c) The process of converting a byte stream to an object.
   d) The process of creating a new object from an existing object.

   Answer: b) The process of analyzing and modifying the runtime behavior of classes and objects.

2. MCQ (Coding): How do you obtain the `Class` object representing a class in Java?

   a) By calling the `getClass()` method on an object of the class.
   b) By calling the `Class.forName("ClassName")` method.
   c) By using the `.class` syntax on the class name, like `ClassName.class`.
   d) All of the above.

   Answer: d) All of the above.

3. MCQ: What information can be obtained from the `Class` object in Java?

   a) Fields and methods of the class.
   b) Constructor information of the class.

c) Interfaces implemented by the class.
d) All of the above.

Answer: d) All of the above.

4. MCQ (Coding): How do you create an instance of a class using reflection in Java?

a) By calling the `newInstance()` method on the `Class` object.
b) By calling the `createInstance()` method on the `Class` object.
c) By using the `new` keyword followed by the `Class` name.
d) By using the `reflect()` method on the `Class` object.

Answer: a) By calling the `newInstance()` method on the `Class` object.

5. MCQ: What is the main advantage of using reflection in Java?

a) It allows access to private fields and methods of a class.
b) It allows bypassing the security checks performed by the JVM.
c) It allows creating new classes at runtime.
d) It allows accessing the memory addresses of objects.

Answer: a) It allows access to private fields and methods of a class.

6. MCQ (Coding): How do you set the value of a private field of a class using reflection in Java?

a) By calling the `setField()` method on the `Class` object.
b) By using the `set()` method on the field object obtained from the `Class` object.
c) By calling the `setAccessible(true)` method on the `Class` object.
d) By using the `setValue()` method on the field object obtained from the `Class` object.

Answer: b) By using the `set()` method on the field object obtained from the `Class` object.

7. MCQ: What is the performance impact of using reflection in Java?

a) It has no performance impact.
b) It can improve performance in some cases.
c) It can significantly degrade performance.
d) It depends on the complexity of the class being reflected.

Answer: c) It can significantly degrade performance.

8. MCQ (Coding): What is the purpose of the `getInterfaces()` method in the `Class` class?

a) It returns an array of all the interfaces implemented by the class.
b) It returns an array of all the superclasses of the class.
c) It returns an array of all the fields declared in the class.
d) It returns an array of all the methods declared in the class.

Answer: a) It returns an array of all the interfaces implemented by the class.